DOCUMENT RESUME

ED 028 811 LI 001 483

By-Artandi, Susan; Baxendale, Stanley

Project MEDICO Third Progress Report.

Rutgers, The State Univ., New Brunswick, N.J. Graduate School of Library Service.
Spons Agency-Public Health Service (DHEW), Washington, D.C. National Library of Medicine.
Report No-PHS-LM-94
Pub Date 69
Note-71p.; The first progress report is ED 022 504 and the second progress report is LI 001 482
EDRS Price MF-$0.50 HC-$3.65
Descriptors-*Automation, Computer Programs, Computers, Computer Storage Devices, Dictionaries, *Indexing, *Information Retrieval, *Information Storage, Input Output, Operations Research, *Search Strategies
Identifiers-MEDICO, *Model Experiment in Drug Indexing by Computer

This report describes the searching methods and the search program for the automatic indexing method which was developed and implemented in an earlie r phase of the project. The indexing method generates index tags automatically from English language text and creates a machine searchable file of index records for the document being processed. Since the First Progress Report the indexing program has been modified to facilitate the updating and expansion of the computer-stored dictionary. The MEDICO file which is the output of the automatic indexing program is a direct file stored on magnetic tape and is sequenced by document accession number. The primary access point of the file can involve as many as four hierarchical levels and generic searches are easily implemented. Boolean searches allow for the retrieval of highly specific information. Prior to searching, the Boolean expressions corresponding to the natural language query are formulated by the human searcher. Normalization of the query to make it compatible with the index language is accomplished automatically by the computer. The tape file is searched sequentially to search for the presence or absence of terms as prescribed in the Boolean expression. Several queries can be processed simultaneously and the output for each query can be printed out as a separate unit. (Author/JW)

ED028811

LI [redacted]
001483

PROJECT MEDICO

Third Progress Report

by
Susan Artandi
and
Stanley Baxendale

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

[redacted]
LI 001483

Graduate School of Library Service
Rutgers, the State University
New Brunswick, New Jersey

1969

Project MEDICO

Third Progress Report

(LM-94 Grant)

by

Susan Artandi

Graduate School of Library Service
Rutgers, the State University

and

Stanley Baxendale

Department of Computer Sciences
Rutgers, the State University

Graduate School of Library Service
Rutgers, the State University
New Brunswick, New Jersey

1969

FOREWORD

The work described in this Third Progress Report was conducted under grant LM-94 from the Public Health Service National Library of Medicine.

Rutgers University personnel participating in this phase of the project are:

Dr. Susan Artandi, Principal Investigator

Associate Professor, Graduate School of Library Service

Mr. Stanley Baxendale

Associate Professor, Department of Computer Sciences

Mrs. Gillian McElroy, MLS

A great deal of advice was received from Dr. Thomas H. Mott, Jr., Director, Center for Computer and Information Services and Chairman, Department of Computer Sciences.

ABSTRACT

The searching method and the search program for the automatic indexing method developed in an earlier phase of the Project is described. The MEDICO file which is the output of the MEDICO automatic indexing program is a direct file on magnetic tape. The primary access points of the file can involve as many as four hierarchical levels. Generic searches are easily implemented and Boolean searches allow the retrieval of highly specific information.

TABLE OF CONTENTS

		Page
I.	Introduction	1
II.	Summary	3
III.	The MEDICO Index File	5
IV.	The Search Program	9
V.	Modifications in the Indexing Program	15
VI.	Appendix	19
	1. Flow Charts	21
	2. Sample Query Input to Search Program and Its Corresponding Output	29
	3. Sample Records from the Automatic Index File	35
	4. Automatic Indexing Program Listing	39
	5. Revised Version of the Dictionary Processing Program	55
	6. Search Program Listing	61
	7. Print Program Listing	71

I. INTRODUCTION

This is the third Progress Report on research in automatic indexing of drug-related information conducted under grant LM-94 from the Public Health Service National Library of Medicine.

The First Progress Report,[1] published in January 1968, describes the automatic indexing method which was developed in the Project. The method will generate index tags automatically from English language text. Pre-defined text characteristics are used to alert the computer to the presence of information that should be indexed. In the process of indexing the computer will switch from the uncontrolled vocabulary of the text to the controlled vocabulary of the index language and it will automatically compute weights for index terms. The indexing algorithm also includes a method for the automatic generation of links.

The Second Progress Report,[2] published in November 1968, is concerned with the statistical evaluation of the output of the MEDICO automatic indexing method just described. The statistical tests were primarily designed to examine the validity of the assumptions which formed the bases of the algorithms developed for the computation of weights and the generation of links between index terms. Evaluation also included a comparison of the output generated from full text and from the processing of abstracts and summaries of the same articles.

---

[1]Artandi, S. and S. Baxendale, Project MEDICO. First Progress Report. New Brunswick, N. J., Graduate School of Library Service, Rutgers, The State University, 1968.

[2]Artandi, S. and E. H. Wolf. The effectiveness of weights and links in automatic indexing. Project MEDICO. Second Progress Report. New Brunswick, N. J., Graduate School of Library Service, Rutgers, The State University, 1968.

II. SUMMARY

This Third Progress Report describes the searching methods and the search program for the automatic indexing method which was developed and implemented in an earlier phase of the Project. The indexing method will generate index tags automatically from English language text and by utilizing explicitly defined text characteristics it creates a machine searchable file of index records for the document being processed. Some modifications in the indexing program are also described.

The MEDICO file which is the output of the automatic indexing program is a direct file stored on magnetic tape and is sequenced by document accession number. The primary access points of the file can involve as many as four hierarchical levels and generic searches are easily implemented. Boolean searches allow for the retrieval of highly specific information.

Prior to searching, the Boolean expressions corresponding to the natural language query are formulated by the human searcher. Normalization of the query to make it compatible with the index language is accomplished automatically by the computer. The tape file is searched sequentially to search for the presence or absence of terms as prescribed in the Boolean expression. Several queries can be processed simultaneously and the output for each query can be printed out as a separate unit.

III. THE MEDICO INDEX FILE

The MEDICO file is a direct file of document references and their associated index terms stored on magnetic tape in accession number order. In this sense the MEDICO file is similar to many other index files. The principal characteristic which distinguishes it from other files is that its content and format is automatically generated by the computer.

The specification of the content of the index record is accomplished through the MEDICO indexing algorithm described in the First Progress Report. The output of the indexing program creates a record for each document containing the following data elements: author, title, citation, and index terms with their respective weights and Chemical Abstracts Registry Numbers.

Since the MEDICO file is a direct file each record stands for a single document as opposed to an inverted file in which each record stands for a single index term. Inherent in the process of searching a direct file for documents specified by subject is the need to make a complete scan of the entire file for each query to be processed. The capability for simultaneous searches, processing several queries in a single pass of the tape, can compensate for this limitation.

The MEDICO search program provides for simultaneous searches and allows for the output corresponding to each query to be printed out separately.

Searching is essentially the reverse of indexing and the preparation of a search instruction involves procedures and sources of errors that are very similar to those encountered in indexing. The objective of searching is to identify those documents whose content is relevant to the query.

The output of a search may be viewed as the result of the relevance judgment of the system. Theoretically, the closer this resembles the relevance judgment of the user the better the system performs. In practice, however, the problem is not quite as clearcut, and factors influencing both system and user judgment need to be taken into consideration.

In searching a documentary file communication with the file is accomplished through the index terms included in the records of the various documents. The nature of this communication

or the flexibility of searching will be to a large extent determined by the contents and the structure of the file records.

The MEDICO record was designed to allow access to the drug information at as many as four hierarchical levels. Figure 1 shows these hierarchical levels.

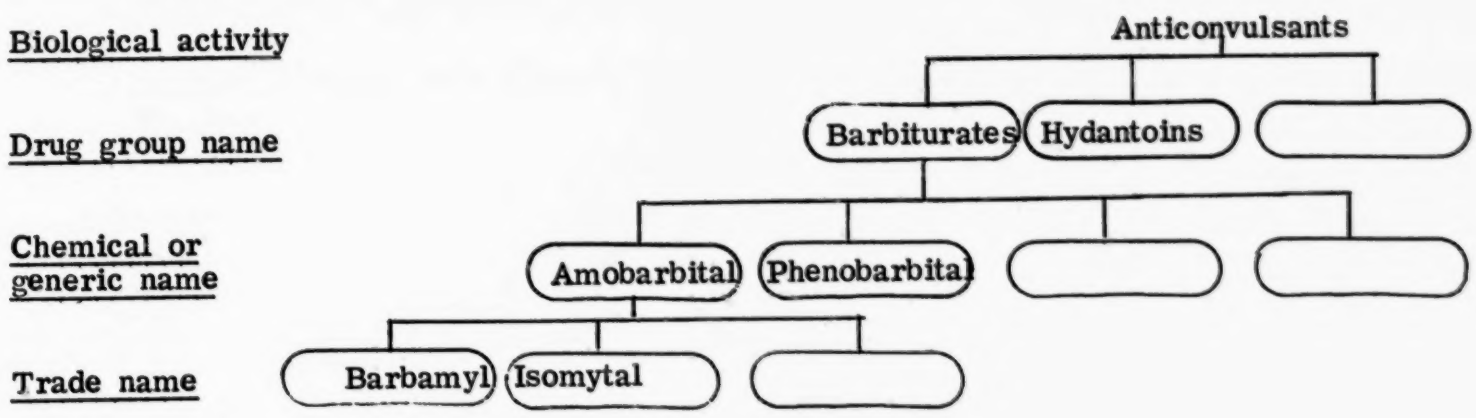


FIG. 1

Since the drug group name is automatically posted for each compound, and since the chemical name and the drug group name are automatically posted for each trade name, generic searches are relatively easy to make. The generic specific relationships are also displayed in the printout of the index record. All this is accomplished through the use of "packages."

As it was already explained in the First Progress Report, a package is associated with each term in the system and it consists of those terms which will appear in the index record whenever that particular term is recognized in the document text.

For example, if the name 5-ethyl-5-isopentylbarbituric acid were to appear in the text of the document, its corresponding package would be recorded in the index record:

amobarbital
5-ethyl-5-isoamylbarbituric acid
barbiturates
Reg. no. 57534

The same package would be generated if 5-ethyl-5-isoamylbarbituric acid, or amobarbital were found in the text. If, however, a trade name such as Barbamyl would appear in the text, the package would also include Barbamyl in addition to what is given above.

It should be noted that the fourth hierarchical level, group name according to chemical activity, was omitted from the MEDICO record because all of the drugs in the experiment were anticonvulsants. Whenever the term anticonvulsants appears in a record it means that it was

generated directly from the text and not through generic posting.

In addition to utilizing the primary access points in the index record, the search program is designed to make possible complex Boolean searches using the connectives AND, OR, and NOT. Any data element can be combined with any other included in the subject description part of the index record, involving the capability for many more possible cominbations than would probably be needed in practice.

While the Boolean expression corresponding to the query is formulated by the human searcher, normalization of the search terms to make the query compatible with the index record is accomplished automatically by the computer. The index file is searched sequentially to search for terms as prescribed in the Boolean expression. Several queries can be searched simultaneously and the output relating to each query can be printed out as a separate unit.

IV. THE SEARCH PROGRAM

After the Boolean expressions have been formulated the program automatically performs the normalization of the query. Normalization means that the program substitutes for the uncontrolled terms in the query the corresponding terms from the system vocabulary.

The file is scanned to find the terms in the query. When a term is present a hit is scored; however, the final selection of an index record as a reference depends on the truth value of other terms in the query and the resultant truth value of the logical expression forming the query. The output of answers for any query will be a list of selected article numbers. When the whole file has been passed these arrays of article numbers are used to control the printout of the index records of articles that satisfy the queries. The main problem at this stage is to process the print tape sequentially and to print out all the records for a query sequentially at the same time, followed by the printout of the records required by the results of the next query and so on. The program here uses as much main storage as possible and uses the Random Access Disk Unit as auxiliary storage.

Queries are punched into cards in the format shown in Fig. 2. Card column 1 contains a letter A, conveniently but not necessarily, the first letter of the term, ANTICONVULSANT. The first twenty-four letters of the alphabet can be used for this purpose, but not Y or Z. These are reserved for the truth values true and false as will be explained in the description of the routine for evaluating logical expressions. Thus twenty-four different terms could appear in one query.

If two or more terms begin with the same letter it would be necessary to use other letters to represent them.

The use of letters permits a useful condensation of the logical form of the query. Card column 3 contains a single numeric digit code which indicates the type of term according to the following table.

Term type codes used on query cards

1 ... Author
2 ... Drug term
3 ... Registry number
4 ... Association linkage (must be two cards together)
5 ... Generic term
6 ... End of query
7 ... End of all queries

Figure 2

Coding of logical form of query

Boolean connectives

* ... AND
... OR
\- ... NOT

The Boolean expression does not have to start in card column 1 and spaces can be used to improve legibility.

As the query cards are read the information they contain is stored in arrays. Each item is associated with the letter representing it in the query and its position in the array. Each logical form of query is also stored. A 7 in a query card signals the end of all queries and causes the first index record to be read from the binary search tape and passes control to the logic evaluation routine.

Searching for a particular query in the index records is simply a case of comparing the query terms successively with the terms in the index records. If the search is successful the truth value is set to true. So that as far as the particular term is concerned a hit has been made.

The basic ideas used in evaluating the logical expression are simple. Consider that there are two kinds of, what we may call, elementary logical expressions, namely, those involving AND and those involving OR. The expression A AND B AND C is false as soon as a false term is found. That is, if A is false the whole expression is false and it is only necessary to fail in the search for A in the index record to be able to abandon any further search. Similarly the expression A OR B OR C can be considered true as soon as a true term is found. Negation just reverses the truth value of a term.

Sample Worksheet for Query Formulation

QUERY

Which drugs (with the exception of barbiturates, hydantoins, and succinimides) have anti-convulsant activities?

CODED QUERY:

CC 1234		WEIGHT Card Col 80
A	4 ANTICONVULSANTS	2
	4 ACTIVITY	
B	2 BARBITURATES	
H	2 HYDANTOINS	3
S	2 SUCCINIMIDES	2
	6 END OF QUERY	

LOGICAL FORM OF QUERY:

$A^{*}-(B^{*}H^{*}S)$

FIG. 3

The MEDICO search program will evaluate the truth value of complicated Boolean expressions, including nested parenthetical expressions.

Parenthetical expressions can be evaluated easily by putting the expression in a working storage area and using it to keep track of events in the following manner.

First a left parenthesis is located and its position noted, next the first right parenthesis is located. Moving to the left find the nearest left parenthesis. This may or may not be the first left parenthesis encountered. At any rate within these parentheses an elementary expression is situated. The logic program calls successively on the search to establish the truth of the terms and the truth of expression is evaluated. Now the left parenthesis and all the terms in the elementary expression under discussion are blanked out and if the expression is true the right parenthesis is replaced by a Y meaning true, otherwise a Z meaning false. Applying this process recursively the truth of the query is established at the earliest opportunity. The result is a very fast search. All of the queries in a batch are processed against each index record in sequence, that is, all the queries are processed against the file in one pass of the tape.

The output from this program is an array of the numbers of articles 'hit' by each query in succession.

The problems involved in searching relates to the sequential nature of a tape file and the relatively limited capacity of main storage. Given the array of hits produced by the query search program it is required to produce a set of printed records in proper sequence for each query in turn. This is accomplished in one pass of the print record tape which contains the index records in ascending order of article number.

At this concluding stage of the program only the print program and its associated subroutines need be in memory so that the search program can be overlaid and there is more room available for storing article records. However, with a large file it would be necessary to use auxiliary disk storage. In brief, index records are read into memory and printed out in answer to the query being processed. If the query needs an article, the articles preceding it on the tape are read into memory until the required article is reached and can be read in to be printed out. Proceeding in this way it is possible to run out of space in memory. To provide room some articles in memory are written out to disk storage and since disk storage is addressable, these articles can easily be read back when required. The articles chosen

for shunting on to disk are those with lowest priority. The traffic control of the movement and printing is by means of the SQUEEZ subroutine.

V. MODIFICATIONS IN THE INDEXING PROGRAM

Some changes were made in the indexing program since the First Progress Report. The modifications are primarily intended to facilitate the updating and expansion of the dictionary. The complete revised indexing program is included in the Appendix.

As it was described in the First Progress Report, associated with dictionary terms in memory is a matrix of transfer addresses. As a text word is checked against the dictionary the first two characters of the word are converted into a unique numeric index. This index is, in effect, an 'indirect address' of a location in the matrix which contains the address of the first term in the dictionary beginning with the same two characters. Suppose the textword is AMOBARBITAL, then automatically the first two letters of the word are used to 'look up' or locate the first term in the dictionary beginning with AM which is AMINOGLUTHEMIDE. However, if the text word were AXILLARY the index deriving from AX would find a location in the matrix of transfer addresses containing a zero indicating that the dictionary did not contain any terms beginning with AX.

Initially, when the dictionary was first set up the terms were sorted according to the IBM 7040 scientific collating sequence using the available system IBSORT routine. Before this modification was made a section of the dictionary containing the words under consideration looked as in the figure below.

542	25	ALPHA(P-AMINOPHENYL)-ALPHA ETHYL-GLUTARIMIDE
547	-25	AMINOGLUTETHIMIDE
552	25	AMINO-GLUTETHIMIDE
556	-9	AMOBARBITAL
561	-14	AMOBARBITAL SODIUM
566	-9	AMYLOBARBITONE
569	9	AMYTAL
574	14	AMYTAL SODIUM
579	-49	ANTICONVULSANT

Any term is preceded by two numeric terms; the first points to the beginning of the next dictionary term and the second is the package number with a sign indicating whether the

term has to be placed in the index record together with the package. If the sign is positive the term will be printed in the record together with the package, if negative, only the package terms are printed.

For example, 542 is the address of the first computer word of AMINOGLUTETHIMIDE. 542 is the address of the first term beginning with AM and consequently 542 appears in the matrix of transfer addresses at the location designated by the index constructed from the letters AM. Thus, its appearance in the dictionary can be considered to be redundant. Its use, however, allowed the calculation of the length of the dictionary term plus two as the difference between two successive addresses. 547-542 gives 5 as the number of computer words containing the following information.

547 -25 AMINOGLUTHETHIMIDE

Hence, the term is contained in 3 computer words. This length computed in the dictionary processing program is thus implicitly available for use in the indexing process and the search program.

The -25 indicates that the associated package is number 25 and that the term does not have to be printed in addition to the package terms because, in this case, it is one of the terms in the package.

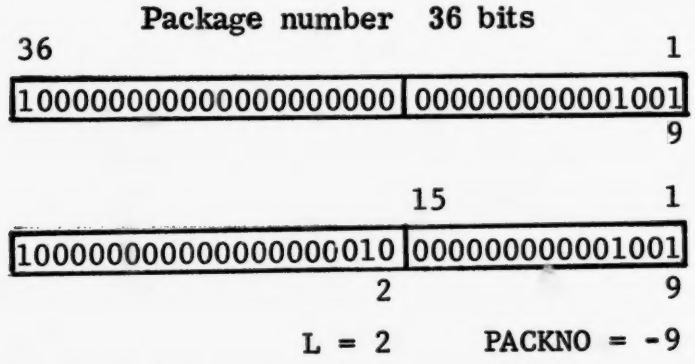
Two modifications were introduced; one, was to replace the address 542 with a zero which is now used to indicate the last term beginning with AL. Similarly, 574 which is the location of the first term beginning with AN can be made 0 to indicate the last term, AMYTAL SODIUM, in the group of terms beginning with AM. Another simple modification allows the storage of the length of the dictionary term in the word containing its package number.

Consider the example in the following figures.

556	-9	AMOBAR	BITAL

The previous address was 552 and hence 556-552=2 gives 2 computer words for the length of AMOBARBITAL.

The package number word is represented below.



Since only four bits are required to represent the length of a term which is not likely to exceed 15 computer words it is feasible to let each address be the link to the next term in that group wherever it is placed instead of calculating length from the addresses of successive terms. This removes any necessity to sort the terms and it is possible to update the dictionary quite freely. The portion of the dictionary described before would now appear as indicated below.

0	8/25	ALPHA(P-AMINOPHENYL)-ALPHA ETHYL-GLUTARIMIDE
547	3/-25	AMINOGLUTETHIMIDE
552	3/25	AMINO-GLUTETHIMIDE
556	2/-9	AMOBARBITAL
561	3/-14	AMOBARBITAL SODIUM
566	3/-9	AMYLOBARBITONE
569	1/9	AMYTAL
0	3/14	AMYTAL SODIUM
579	3/-49	ANTICONVULSANT

Trying to match AMOBARBITAL would result in the length 2 being compared successively with the lengths 3, 3 and finding equality of length at the third attempt. Following the successful length test, and the testing of the successive computer words from which the term is made up AMOBARBITAL will give a hit. If the word being tested were AMMONIA the length test would be successful at AMOBARBITAL since each term occupies two computer words; however, the comparison of computer words would fail on the first comparison. Subsequent length tests would fail including the test against the length of AMYTAL SODIUM. The search is now terminated with no hit since AMYTAL SODIUM is the last term in the group as signalled by the zero.

With the modification just described new terms can be added to the dictionary without

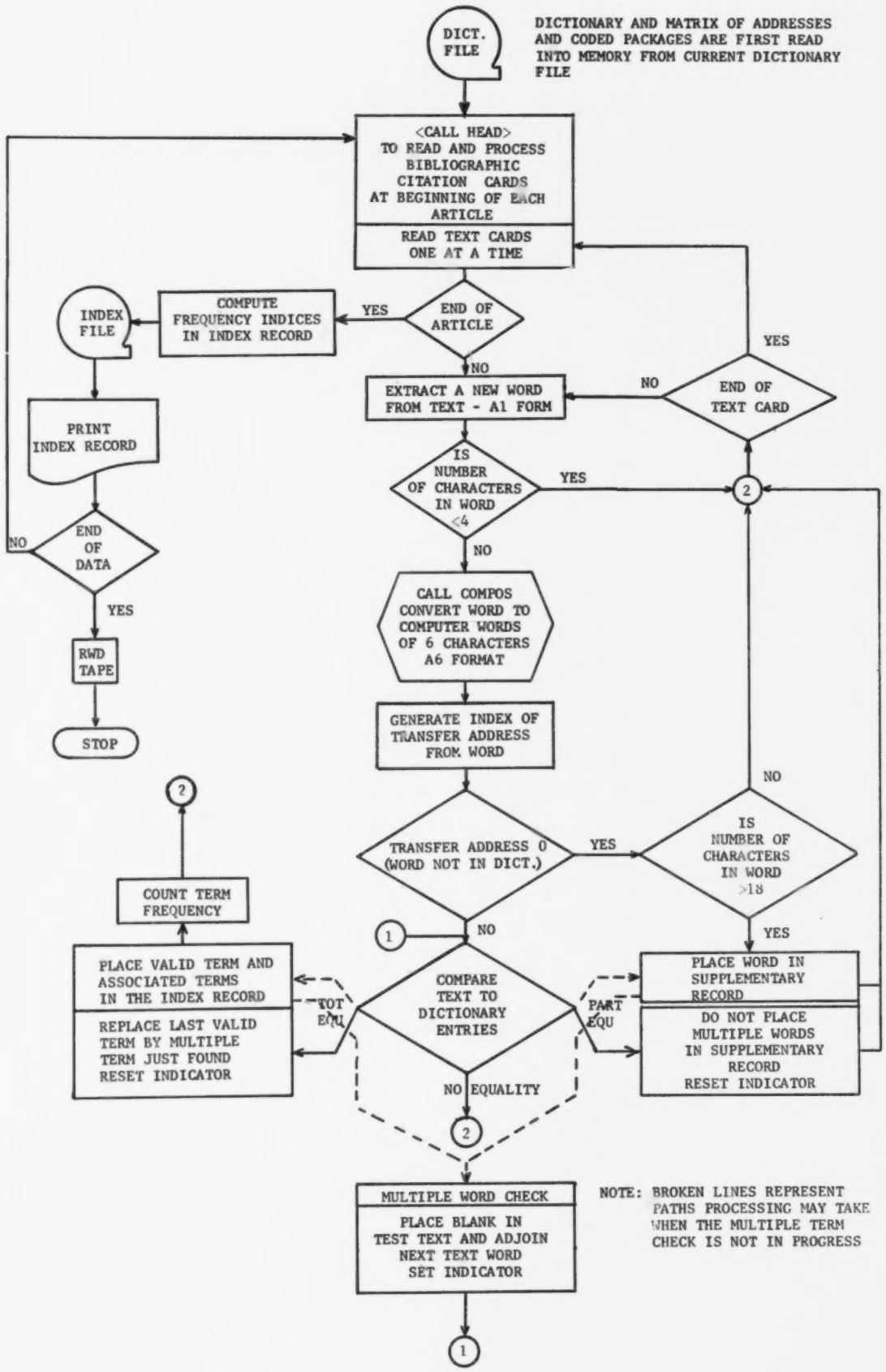
invalidating the content of previously created index records. When new terms are added to the dictionary they are simply placed at the end and the pointer to the new term is placed in the link word of the previously last word in the group replacing the zero denoting the end of the string. Any number of new terms can be added and linked in the usual way with the last word in a group being designated by the zero address.

If it is necessary to add a term, for example, AXILLARY and no group of terms of which the first two letters are AX exists in the dictionary then its computed index will point to a location in the transfer address matrix which contains a zero. The program inserts the address of the available location in the dictionary at which the term will be placed into the appropriate word of the transfer matrix replacing the zero.

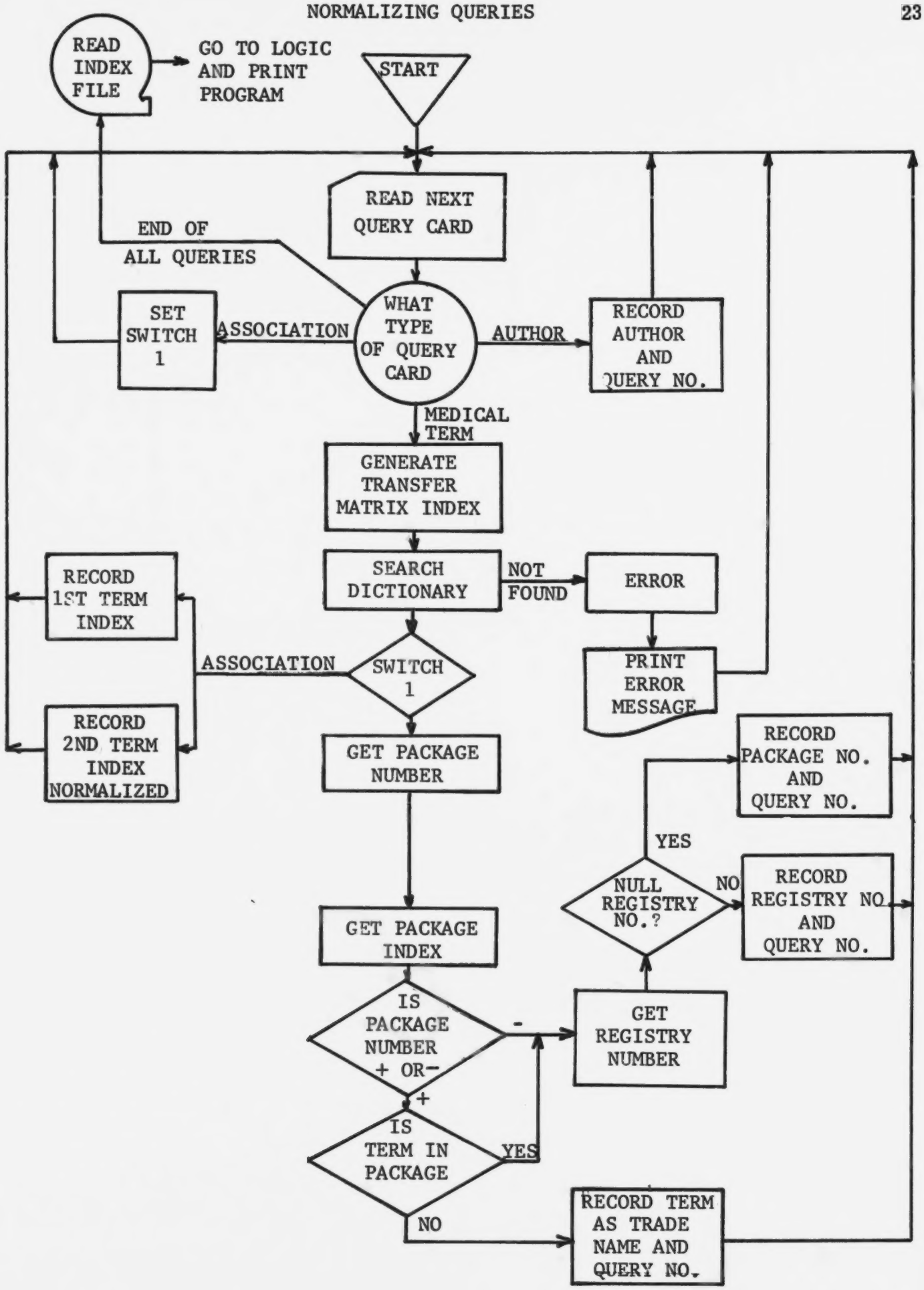
APPENDIX

1. Flow Charts

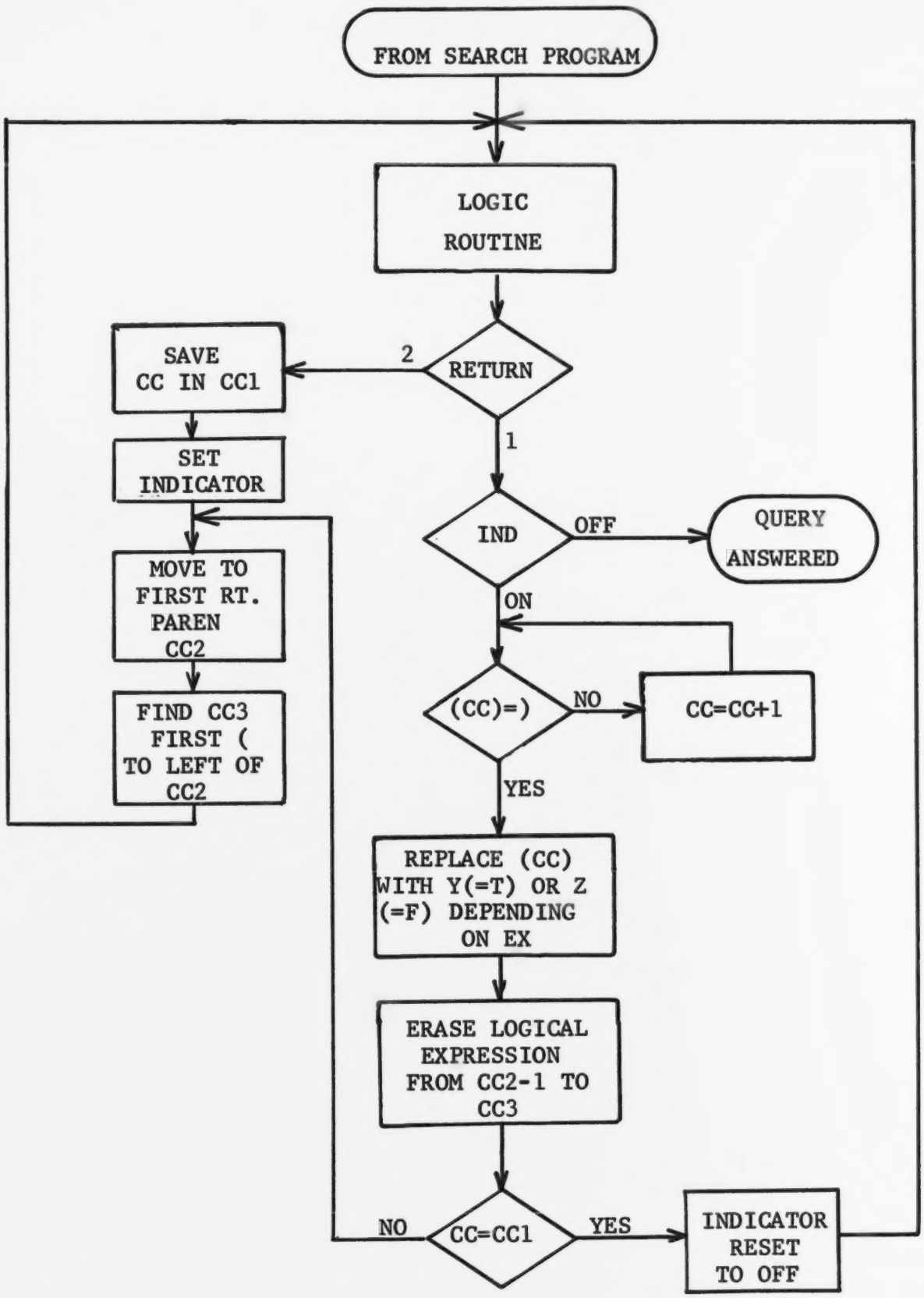
FLOW CHART OF AUTOMATIC INDEXING PROGRAM


DICT. FILE
DICTIONARY AND MATRIX OF ADDRESSES AND CODED PACKAGES ARE FIRST READ INTO MEMORY FROM CURRENT DICTIONARY FILE
<CALL HEAD> TO READ AND PROCESS BIBLIOGRAPHIC CITATION CARDS AT BEGINNING OF EACH ARTICLE
READ TEXT CARDS ONE AT A TIME
END OF ARTICLE
YES
COMPUTE FREQUENCY INDICES IN INDEX RECORD
INDEX FILE
NO
EXTRACT A NEW WORD FROM TEXT - A1 FORM
NO
END OF TEXT CARD
YES
PRINT INDEX RECORD
IS NUMBER OF CHARACTERS IN WORD <4
YES
2
NO
END OF DATA
NO
YES
CALL COMPOS CONVERT WORD TO COMPUTER WORDS OF 6 CHARACTERS A6 FORMAT
RWD TAPE
STOP
GENERATE INDEX OF TRANSFER ADDRESS FROM WORD
NO
TRANSFER ADDRESS 0 (WORD NOT IN DICT.)
YES
IS NUMBER OF CHARACTERS IN WORD >18
2
COUNT TERM FREQUENCY
1
NO
YES
PLACE VALID TERM AND ASSOCIATED TERMS IN THE INDEX RECORD
REPLACE LAST VALID TERM BY MULTIPLE TERM JUST FOUND RESET INDICATOR
TOT EQU
COMPARE TEXT TO DICTIONARY ENTRIES
PART EQU
PLACE WORD IN SUPPLEMENTARY RECORD
DO NOT PLACE MULTIPLE WORDS IN SUPPLEMENTARY RECORD RESET INDICATOR
NO EQUALITY
2
MULTIPLE WORD CHECK
PLACE BLANK IN TEST TEXT AND ADJOIN NEXT TEXT WORD SET INDICATOR
1

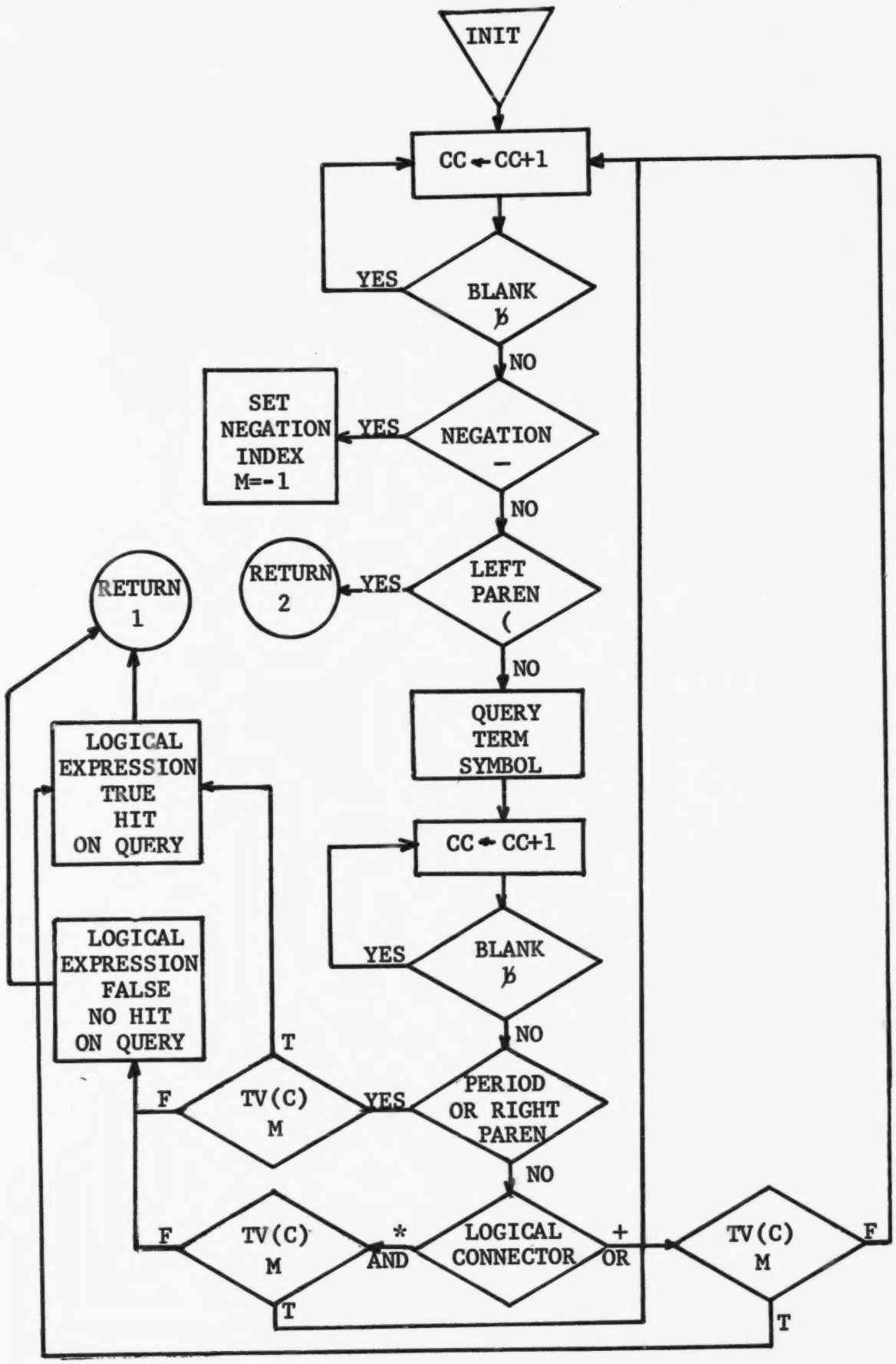
NOTE: BROKEN LINES REPRESENT PATHS PROCESSING MAY TAKE WHEN THE MULTIPLE TERM CHECK IS NOT IN PROGRESS


READ INDEX FILE
GO TO LOGIC AND PRINT PROGRAM
START
READ NEXT QUERY CARD
END OF ALL QUERIES
SET SWITCH 1
ASSOCIATION
WHAT TYPE OF QUERY CARD
AUTHOR
RECORD AUTHOR AND QUERY NO.
MEDICAL TERM
GENERATE TRANSFER MATRIX INDEX
SEARCH DICTIONARY
NOT FOUND
ERROR
RECORD 1ST TERM INDEX
ASSOCIATION
SWITCH 1
PRINT ERROR MESSAGE
RECORD 2ND TERM INDEX NORMALIZED
GET PACKAGE NUMBER
RECORD PACKAGE NO. AND QUERY NO.
YES
NULL REGISTRY NO.?
NO
RECORD REGISTRY NO AND QUERY NO.
GET PACKAGE INDEX
IS PACKAGE NUMBER + OR−
-
GET REGISTRY NUMBER
+
IS TERM IN PACKAGE
YES
NO
RECORD TERM AS TRADE NAME AND QUERY NO.

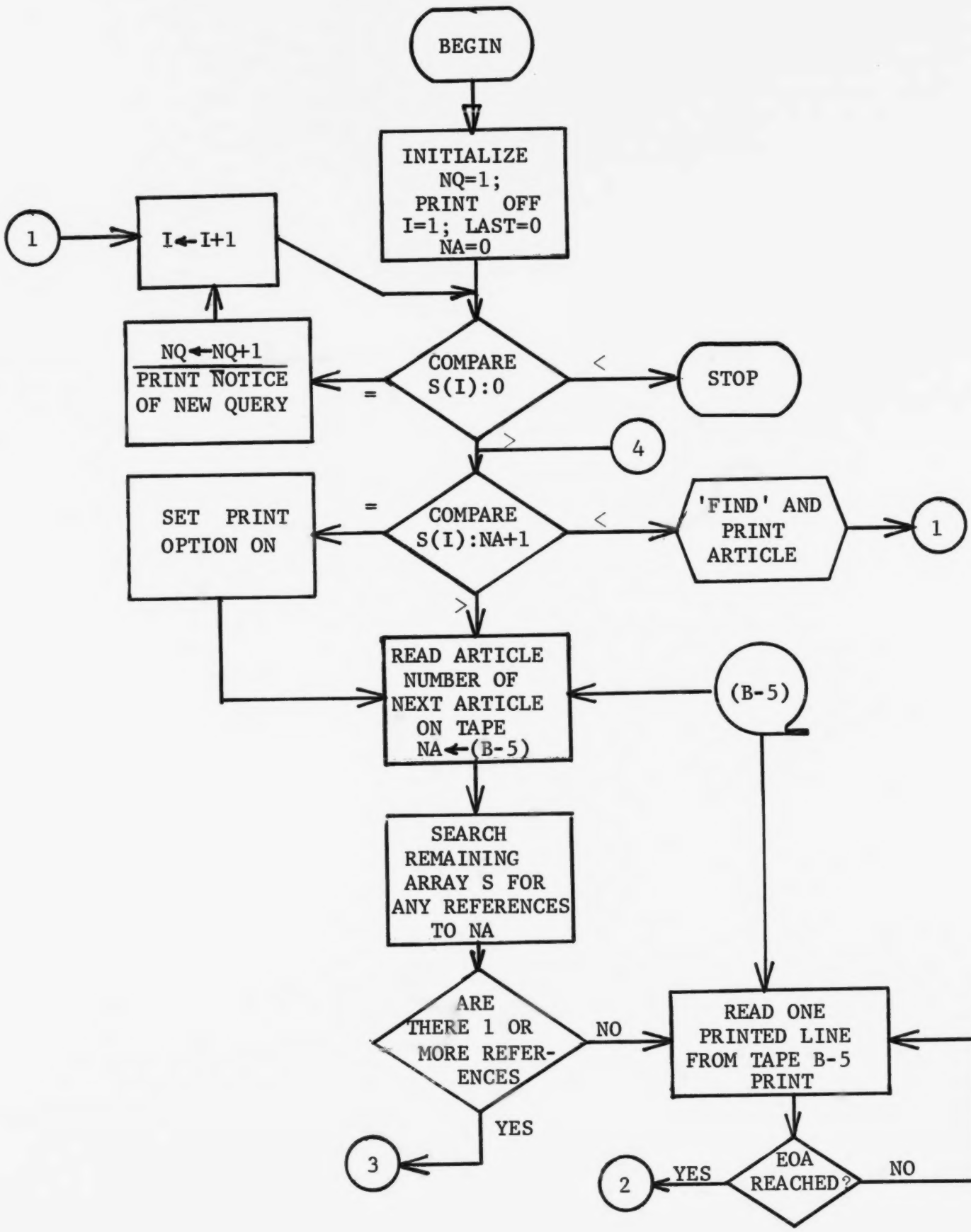
LOGIC PROGRAM



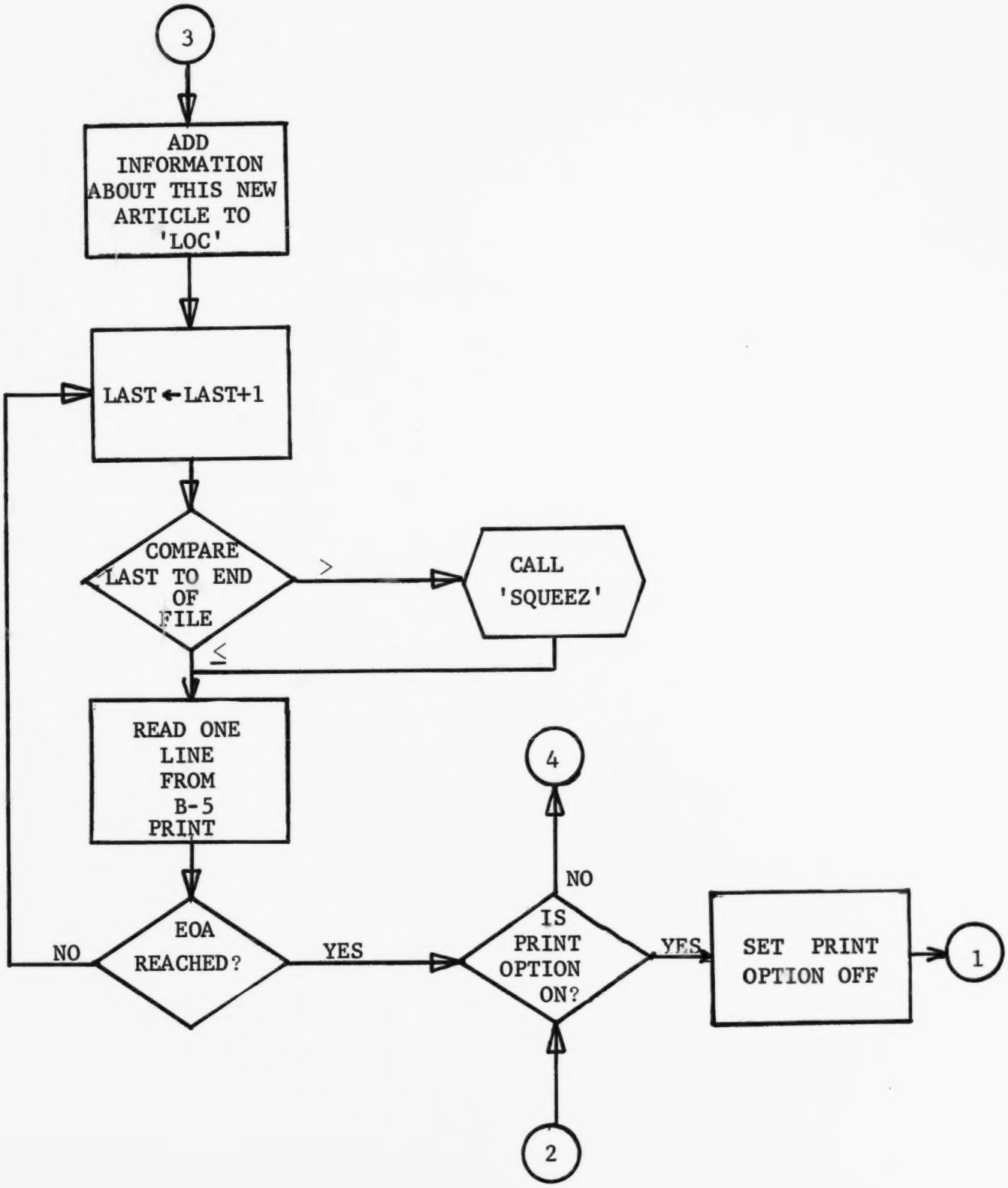
LOGIC SUBROUTINE



PRINT PROGRAM - CHART I

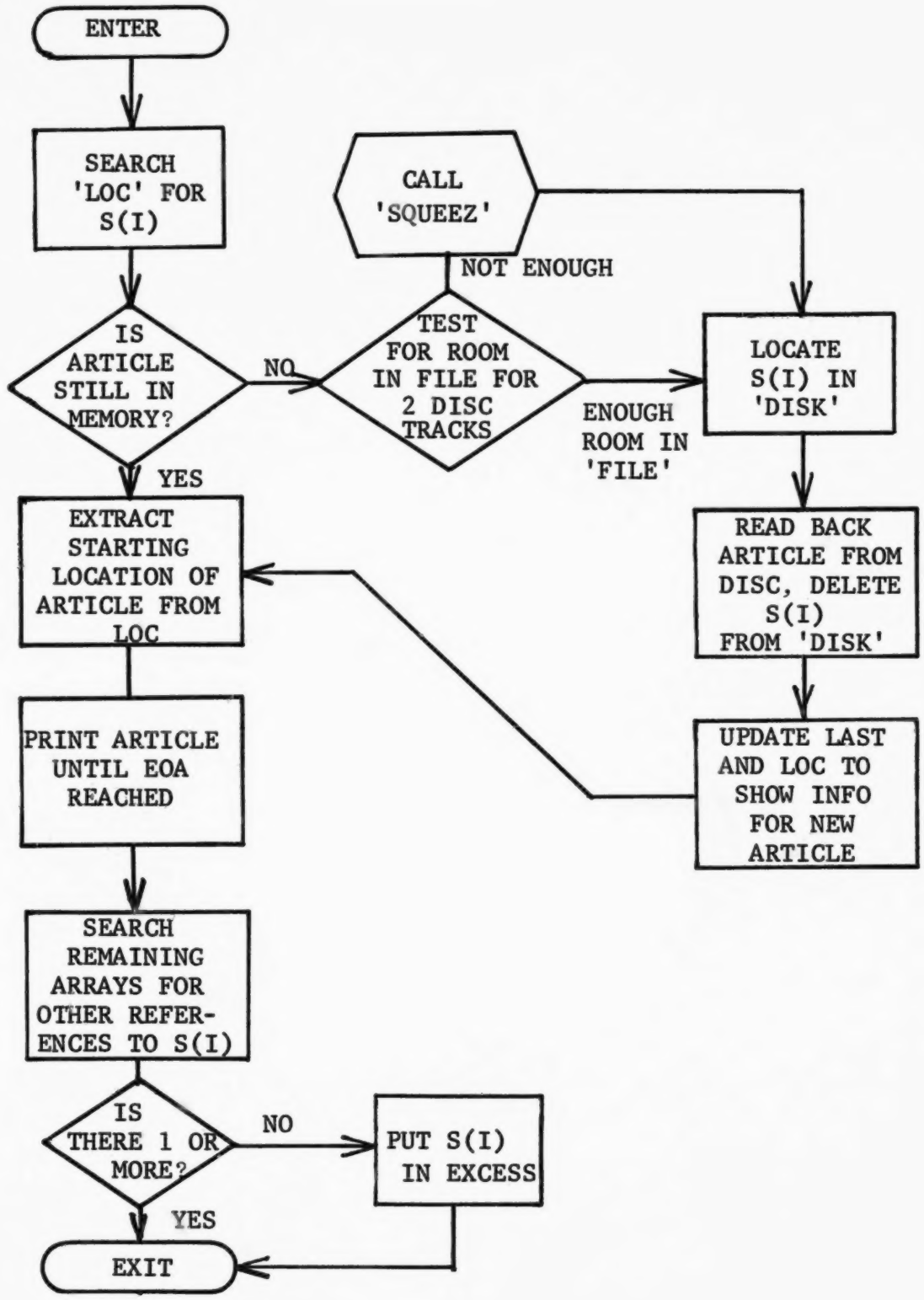


PRINT PROGRAM - CHART 2



PRINT PROGRAM - CHART 3

EXPANSION OF 'FIND' AND PRINT ARTICLE


ENTER
SEARCH 'LOC' FOR S(I)
IS ARTICLE STILL IN MEMORY?
NO
YES
CALL 'SQUEEZ'
NOT ENOUGH
TEST FOR ROOM IN FILE FOR 2 DISC TRACKS
ENOUGH ROOM IN 'FILE'
LOCATE S(I) IN 'DISK'
READ BACK ARTICLE FROM DISC, DELETE S(I) FROM 'DISK'
UPDATE LAST AND LOC TO SHOW INFO FOR NEW ARTICLE
EXTRACT STARTING LOCATION OF ARTICLE FROM LOC
PRINT ARTICLE UNTIL EOA REACHED
SEARCH REMAINING ARRAYS FOR OTHER REFER-ENCES TO S(I)
IS THERE 1 OR MORE?
NO
PUT S(I) IN EXCESS
YES
EXIT

2. Sample Query Input to Search Program and its Corresponding Output

Query 1. Drugs which are active as anticonvulsants but which are not of the barbiturate, hydantoin or succinimide groups.

```
A 4ANTICONVULSANTS
  4ACTIVITY
B 2BARBITURATES
H 2HYDANTOINS
S 2SUCCINIMIDES
  6
A * -(B+H+S).
```

Query 2. The use of 3, 5, 5-trimethyloxazolidine-2, 4-dione as an anticonvulsant.

```
T 23,5,5-TRIMETHYLOXAZOLIDINE-2,4-DIONE
A 2ANTICONVULSANT
  6
T * A.
```

Query 3. The use of barbiturates in the treatment of convulsive disorders with the exception of amobarbital.

```
A 4ANTICONVULSANTS
  4TREATMENT
B 2BARBITURATES
P 2AMOBARBITAL
  6
A * B *-P.
```

Query 4. What is the chemical name of sulthiame?

```
S 2SULTHIAME
  6
S.
```

Query 5. What hydantoins other than phethenylate can be used in anticonvulsant therapy?

```
A 4ANTICONVULSANTS
  4THERAPY
H 2HYDANTOINS
P 2PHETHENYLATE
  6
A * H * -P.
```

Query 6. What kind of toxic side effects can be expected when Tegretol is administered?

```
T 4TEGRETOL
  4ADMINISTRATION
U 4TEGRETOL
  4TOXIC
S 4TEGRETOL
  4SIDE EFFECTS
  6
T*U*S.
```

Query 7. Articles on the effectiveness of diphenylhydantoin as a therapeutic agent and articles on recommended dosage.

```
A 4DIPHENYLHYDANTOIN
  4EFFECT
B 4DIPHENYLHYDANTOIN
  4DOSAGE
C 4DIPHENYLHYDANTOIN
  4THERAPY
  6
(A * B) + C.
```

Query 8. The optimum dose of trimethadione in the treatment of epileptics.

```
M 4TRIMETHADIONE
  4DOSE
N 4TRIMETHADIONE
  4EPILEPSY
O 4TRIMETHADIONE
  4THERAPY
  6
M * N * O.
```

Query 9. Administration of diphenylhydantoin sodium in the treatment of trigeminal neuralgia.

```
A 4DIPHENYLHYDANTOIN SODIUM
  4ADMINISTRATION
B 4DIPHENYLHYDANTOIN SODIUM
  4TREATMENT
  6
A * B.
```

Query 10. Articles which have as a central topic the use of primidone as an anticonvulsant. Primidone with weight of 3 and anticonvulsant with weight of 3.

```
P 2PRIMIDONE                                    3
A 2ANTICONVULSANTS                              3
  6
P * A.
```

Query 11. Articles which have as a central topic the use of primidone as an anticonvulsant. Primidone with a weight of 3 and anticonvulsant with a weight of 2.

```
P 2PRIMIDONE                                    3
A 2ANTICONVULSANTS                              2
  6
P * A.
```

Query 12. Articles which refer to oxazolidinediones.

```
O 5OXAZOLIDINEDIONES
  6
O.
```

Query 13. Articles which mention barbiturates.

```
B 5BARBITURATES
  6
B.
```

Query 14. **The chemical name and C.A. registry number of succinimides which are used as anticonvulsants.**

```
S 5SUCCINIMIDES
A 2ANTICONVULSANTS
  6
S * A.
```

QUERY 1 SATISFIED BY ARTICLES 6 13 18 19 20 21 29 30

QUERY 2 SATISFIED BY ARTICLES 4 13 17 21 30

QUERY 3 SATISFIED BY ARTICLES 1 4

QUERY 4 SATISFIED BY ARTICLES 17 28

QUERY 5 SATISFIED BY ARTICLES 4

QUERY 6 SATISFIED BY ARTICLES 0

QUERY 7 SATISFIED BY ARTICLES 2 4 9 11 17 24 26 29
30

QUERY 8 SATISFIED BY ARTICLES 22

QUERY 9 SATISFIED BY ARTICLES 14

QUERY 10 SATISFIED BY ARTICLES 17

QUERY 11 SATISFIED BY ARTICLES 17

QUERY 12 SATISFIED BY ARTICLES 4 13 17 21 22 30

QUERY 13 SATISFIED BY ARTICLES 1 2 3 4 5 6 7 8
9 10 11 13 17 20 21 23
24 26 27 28 29 30

QUERY 14 SATISFIED BY ARTICLES 4 6 17 30

3. Sample Records from Automatic Index File

36

ARTICLE NUMBER 2
BERNSTEIN ZL
TREATMENT OF BARBITURATE COMA.
NEW YORK J MED 66, 2290-4, 1 SEP 66

2428 WORDS IN THIS ARTICLE.

57410 (2) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE,
HYDANTOINS

(3) BARBITURATES

57432 (1) AMOBARBITAL, 5-ETHYL-5-ISOAMYLBARBITURIC ACID, BARBITURATES

50066 (1) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

DIPHENYLHYDANTOIN/ THERAPY (1)
BARBITURATES/ EFFECT (1), ACTIVITY (1), THERAPY (3), DOSAGE (1)

END OF ARTICLE

ARTICLE NUMBER 6
MILLICHAP JG, ORTIZ WR
NITRAZEPAM IN MYOCLONIC EPILEPSIES
AMER J DIS CHILD, 112, 242-248, SEPT. 1966.

2534 WORDS IN THIS ARTICLE.

(3) ANTICONVULSANTS

125337 (1) PRIMIDONE, 5-PHENYL-5-ETHYLHEXAHYDROPYRIMIDINE-4,6-DIONE, BARBITURATES

57410 (1) DIPHENYLHYDANTOIN, 5,5-DIPHENYL-2,4-IMIDAZOLIDINEDIONE, HYDANTOINS

50066 (1) 5-ETHYL-5-PHENYLBARBITURIC ACID, PHENOBARBITAL, BARBITURATES

115388 (1) MEPHOBARBITAL, 5-ETHYL-1-METHYL-5-PHENYLBARBITURIC ACID, BARBITURATES

77418 (1) CELONTIN, N,2-DIMETHYL-2-PHENYLSUCCINIMIDE, SUCCINIMIDES, METHSUXIMIDE

ANTICONVULSANTS/ THERAPY (1), EFFECT (3), ACTIVITY (1)

END OF ARTICLE

4. Automatic Indexing Program Listing

571 BAXEND FORTRAN SOURCE LIST

```
ISN       SOURCE STATEMENT

  0 $IBFTC EDIT
  1       DIMENSION INBUF(72)
  2       DIMENSION NMWORD(50,2)
  3       DIMENSION INDIC(3)
  4       DIMENSION EOA(4)
  5       DIMENSION NOWRD(4),NOLINE(12)
  6       DATA NOWRD,NOLINE/23H WORDS IN THIS ARTICLE.,72H
         1
  7       DATA EOA/22H      *END OF ARTICLE*/
 10       DATA BLANK,Z,COMMA,PERIOD/1H ,1HZ,1H,,1H./
 11       DATA INDIC/1H1,1H2,1H3/
 12       DATA MA,X,DASH,LP,RP,SLASH,AST,EIGHT/1HM,1HX,1H-,1H(,1H),1H/,1H*,
         11H8/
 13       DATA DOUBL,TREBL/0606000000000,0606060000000/
 14       COMMON /H/ MARGIN,ARTNUM,NA,AUTHOR(6,5)
 15       COMMON /LIN/ LINE(132),LINDEX,LMARG,DICT(3000)
 16       COMMON KWORD(14),NWRDS,WORD(72),LENGTH,IEND
 17       EQUIVALENCE (WRDCT,RDCT)
 20       INTEGER DICT
 21       INTEGER PACK4(350),PACK1(100)
 22       INTEGER COLUMN,PERIOD,COMMA,Z,BLANK,WORD,TRADD(4096)
 23       INTEGER GOODWD(100,2),GINDEX
 24       INTEGER SAVE,SVINDX,SUPL(750)
 25       INTEGER SVLNGH
 26       INTEGER SINDX2
 27       INTEGER SUPL2(400)
 30       INTEGER PINDEX,PACKNO,REGNO
 31       INTEGER ENDSEN
 32       INTEGER SUPL3(50,3),TYPE1,TYPE2,SINDX3
 33       INTEGER SW1,SW2
 34       INTEGER WRDCT, WDCT
 35       INTEGER PTERM
 36       INTEGER RP,X,DASH,EIGHT,SLASH,AST
 37       INTEGER PCOUNT
    C     -*-*-*-*-*-*-*-*-*-*
 40       INTEGER ARTNUM,REGNOS(30),PAKNOS(30),TRNAMS(30),COMPAR(100,2)
 41       INTEGER AN
 42       INTEGER REGNWT(30),PKNOWT(30),COMPWT(100)
    C     -*-*-*-*-*-*-*-*-*-*
    C ******************************************************
    C ****** DATA INPUT AND INITIALIZATION SECTION ******
    C ******************************************************
    C READ IN THE PARAMETER CARD.
    C IT IS REQUIRED TO HAVE NUMBERS PUNCHED IN COLUMNS 1-3, 5, 7, AND 8-10
    C OF THE PARAMETER CARD (THE FIRST CARD TO BE READ).
    C THE THREE DIGIT NUMBER PUNCHED IN COLS. 1-3 IS USED BY THE SUBROUTINE
    C    -HEAD- FOR THE MARGIN WIDTH OF THE HEADING OF EACH ARTICLE.
    C A NUMBER PUNCHED IN COLUMN 5 HAS THE FOLLOWING MEANING
    C         1...LIST COMPARISONS AND LARGE WORDS.
    C         2...LIST COMPARISONS ONLY.
    C         3...LIST LARGE WORDS ONLY.
    C         4...DO NOT LIST COMPARISONS OR LARGE WORDS.
    C A NUMBER PUNCHED IN COLUMN 7 HAS THE FOLLOWING MEANING
    C         1...SAVE NON-MEDICAL TERMS.
    C         2...DO NOT SAVE NON-MEDICAL TERMS. (THEIR ASSOCIATION WITH
```

```
571                    BAXEND                    FORTRAN SOURCE LIST EDIT
      ISN       SOURCE STATEMENT

          C              MEDICAL TERMS WILL BE SAVED).
          C THE THREE DIGIT NUMBER PUNCHED IN COLS. 8-10 IS FOR USE BY THE
          C   SUBROUTINE -WRDPJT- FOR THE WIDTH OF THE PRINT LINE USED FOR
          C   OUTPUT OF VALID MEDICAL WORDS, NON-MEDICAL WORDS, AND ASSOCIATIONS.
          C A NUMBER PUNCHED IN COLUMN 11 HAS THE FOLLOWING MEANING
          C     1... PROCESS INDEX FILE FROM BEGINNING
          C     ANY PTHER DIGIT ... ADD RECORDS OF NEW ARTICLES AFTER LAST ARTICLE
          C RECORD ON FILE
          C MODIFICATIONS TO PROGRAM ADDED SINCE PUBLICATION OF PROJECT MEDICO -
          C FIRST PROJECT REPORT - JANUARY 1968 ARE INDICATED BY BROKEN LINES
          C     -*-*-*-*-*-*-*-*-*
           43       READ (5,1) MARGIN,SW1,SW2,LMARG,ARTNUM
           51     1 FORMAT(I3,2I2,I3,I1)
           52       REWIND 4
           53       IF(ARTNUM.NE.1) GO TO 82
           56       REWIND 2
           57       REWIND 3
          C READ IN TRANSFER ADDRESS ARRAY FROM TAPE 8-3.
           60       READ (2) TRADD
           62       WRITE (3) TRADD
          C READ IN DICTIONARY SIZE AND CONDENSED DICTIONARY.
           63       READ (2) J,(DICT(I),I=1,J)
           71       WRITE (3) J,(DICT(I),I=1,J)
          C READ IN PACKAGE ARRAYS.
           76       READ(2) PACK4,PACK1
          101       WRITE (3) PACK4,PACK1
          102       REWIND 2
          103       GO TO 2
          104    82 REWIND 3
          105       READ (3) TRADD
          107       READ(3) J,(DICT(I),I=1,J)
          115       READ (3) PACK4,PACK1
          120    83 READ (3) ARTNUM,AN
          123       IF(AN.NE.999) GO TO 83
          C PROCESS THE BIBLIOGRAPHIC CITATION FOR THE NEXT ARTICLE.
          126     2 CALL HEAD
          127       GO TO (98,97),IEND
          130    97 AN=999
          131       WRITE (3) ARTNUM,AN
          132       CALL EXIT
          C INITIALIZE LENGTH
          133    98 LENGTH = 0
          C INITIALIZE WORD COUNT TO 0.
          134       WRDCT = 0
          C SET -VALID WORD- INDICATOR TO 0 (OFF).
          135       IVALID=0
          C SET -PREVIOUS VALID WORD IN SENTENCE- INDICATOR TO 0 (OFF).
          136       ISENT=0
          C SET -REACHED END OF SENTENCE- INDICATOR TO 0 (OFF).
          137       ENDSEN=0
          C INITIALIZE INDEXES FOR 5 FILES.
          140       GINDEX=1
          141       NMINDX = 1
          142       SVINDX=1
          143       SINDX2=1
```

571 BAXEND FORTRAN SOURCE LIST EDIT

```
ISN       SOURCE STATEMENT

144         SINDX3=1
        C SET FIRST WORD IN EACH FILE TO 0 (THIS IDENTIFIES THE -END OF FILE-
        C     FOR EACH FILE).
145         GOODWD(1,1)=0
146         NMWORD(1,1) = 0
147         SUPL(1)=0
150         SUPL2(1)=0
151         SUPL3(1,1)=0
152         SUPL3(1,2)=0
        C       -*-*-*-*-*-*-*-*-*
153         NR = 0
154         NP = 0
155         NTN = 0
156         REGNOS(1) = 0
157         PAKNOS(1) = 0
160         TRNAMS(1) = 0
        C ***************************************
        C ****** WORD EXTRACTION SECTION ******
        C ***************************************
        C READ IN ONE CARD OF THE TEXT.
161       3 READ(15,4) INBUF
163       4 FORMAT (72A1)
        C CHECK FOR END OF ARTICLE CARD (ZZZ)
164         IF (INBUF(1).EQ.Z.AND.INBUF(2).EQ.Z.AND.INBUF(3).EQ.Z) GO TO 44
        C INITIALIZE TO CHECK BEGINNING OF INPUT BUFFER.
167         DO 5 I=1,72
        C LOOK FOR A BLANK BETWEEN TWO WORDS.
170         IF (INBUF(I).EQ.BLANK) GO TO 6
173         LENGTH=LENGTH+1
        C REMOVE THE NON-BLANK CHARACTER FROM THE INPUT BUFFER.
174         WORD(LENGTH)=INBUF(I)
175       5 CONTINUE
177         GO TO 3
        C CHECK TO SEE IF WORD IS NULL.
200       6 IF(LENGTH.EQ.0) GO TO 5
        C INCREASE THE WORD COUNT.
203         WRDCT = WRDCT+1
204         WDCT = WRDCT
        C WORD IS NOT NULL, CHECK FOR PUNCTUATION AND END OF SENTENCE.
205         IF (WORD(LENGTH).EQ.PERIOD) ENDSEN=1
210         IF (WORD(LENGTH).EQ.PERIOD.OR.WORD(LENGTH).EQ.COMMA) GO TO 7
        C THERE IS NO PUNCTUATION
213         GO TO 8
        C REMOVE LAST CHARACTER (PUNCTUATION)
214       7 WORD(LENGTH)=BLANK
215         LENGTH=LENGTH-1
        C CHECK FOR A SHORT WORD
216       8 IF(LENGTH.LT.4) GO TO 25
        C CONDENSE THE TEXT WORD TO A6 FORMAT.
221         CALL COMPOS
        C ***************************************
        C ****** DICTIONARY SEARCH SECTION ******
        C ***************************************
        C GENERATE TRANSFER MATRIX ADDRESS INDEX
222         IF (KWORD(1).LT.0) GO TO 9
```

```
571                    BAXEND                    FORTRAN SOURCE LIST EDIT
      ISN       SOURCE STATEMENT

      225          INDEX=KWORD(1)/2**24
      226          GO TO 10
      227        9 INDEX=2**11-KWORD(1)/2**24
      230       10 INDEX=INDEX+1
      231          M = TRADD(INDEX)
          C IF TRANSFER ADDRESS IS 0, WORD IS NO GOOD
      232       11 IF (M.EQ.0) GO TO 22
          C IF WORD IS ALPHABETICALLY BELOW DICTIONARY ENTRY, END DICTIONARY
          C SEARCH
      235          IF (KWORD(1)) 12,25,13
      236       12 IF (KWORD(1).GT.DICT(M)) GO TO 19
      241          GO TO 14
      242       13 IF (DICT(M).GT.KWORD(1)) GO TO 19
      245       14 MTEMP=M
          C CHECK FOR A MAXIMUM COMPARISON WITH A DICTIONARY WORD
      246          DO 15 MAXCOM=1,NWRDS
      247          IF (DICT(MTEMP).NE.KWORD(MAXCOM)) GO TO 18
      252       15 MTEMP=MTEMP+1
          C TEST TEXT WORD AND DICT WORD FOR EQUAL LENGTH, IF SAME ... GOOD WORD
      254          IF (NWRDS.EQ.(DICT(M-2)-(M+2)))GO TO 28
      257          SAVE=M
      260          MAX=1
          C MODIFY INDEX M UP TO NEXT DICT ENTRY.
      261       16 M=DICT(M-2)
      262          GO TO 11
          C ********************************************
          C ****** MULTIPLE WORD CHECK SECTION *******
          C ********************************************
          C INCREASE LENGTH AND INDEX I
      263       17 LENGTH = LENGTH+1
      264          I=I+1
          C PUT IN A BLANK
      265          WORD(LENGTH)=BLANK
      266          SVLNGH=LENGTH+1
          C SET INDICATOR TO SHOW THAT A MULTIPLE WORD CHECK IS IN PROGRESS.
      267          MULWRD=1
          C GO BACK TO WORD EXTRACTION SECTION
      270          GO TO 5
          C ************************************************************
          C ****** SAVE WORD SECTION (PARTIAL COMPARISON) ******
          C ************************************************************
          C LOOK FOR PARTIAL COMPARISON, INDEX MAXCOM=1 MEANS NO COMPARISON
      271       18 IF ((MAXCOM.EQ.1).OR.(MAXCOM.LT.MAX+1)) GO TO 16
          C SET SAVE INDICATOR TO DICTIONARY INDEX
      274          SAVE = M
          C SET MAX TO MAXIMUM COMPARISON
      275          MAX=MAXCOM-1
      276          GO TO 16
          C TEST SAVE INDICATOR FOR A NON-VALID WORD, IF ZERO, DO NOT SAVE
          C GO TEST FOR LENGTH
      277       19 IF (SAVE.EQ.0) GO TO 22
          C IF MULTIPLE WORD CHECK IS IN PROGRESS DO NOT SAVE ANY PARTIAL COMPARISONS
      302          IF (MULWRD.EQ.1) GO TO 25
          C WE NOW HAVE A SINGLE WORD THAT COMPARES TO A WORD IN THE DICTIONARY.
          C TEST PROGRAM PARAMETER TO SEE IF THIS WORD SHOULD BE SAVED.
```

```
571                    BAXEND                    FORTRAN SOURCE LIST EDIT
      ISN        SOURCE STATEMENT

      305            GO TO (20,20,17,17), SW1
          C PUT DICTIONARY INDEX, MAXIMUM COMPARISON, WORD LENGTH, AND TEXT WORD
          C IN SUPPLEMENTARY RECORD
      306         20 SUPL(SVINDX) = SAVE
      307            SUPL(SVINDX+1)=MAX
      310            SUPL(SVINDX+2)=NWRDS
      311            SVINDX=SVINDX+3
      312            DO 21 J=1,NWRDS
      313            SUPL(SVINDX)=KWORD(J)
      314         21 SVINDX=SVINDX+1
      316            SUPL(SVINDX)=0
          C ZERO WILL BE WRITTEN OVER BY NEXT ENTRY IN SUPL. RECORD, OTHERWISE
          C IT IS A SIGNAL FOR THE END OF THE SUPL. RECORD
      317            GO TO 17
          C ***********************************************
          C ****** SAVE WORD SECTION (LARGE WORD) ******
          C ***********************************************
          C TEST FOR LARGE SINGLE WORDS.
      320         22 IF ((LENGTH.LT.18).OR.(MULWRD.EQ.1)) GO TO 25
          C PUNCH A CARD WITH THE LONG WORD.
      323            WRITE (7,80) (KWORD(J),J=1,NWRDS)
          C WE NOW HAVE A LARGE WORD (18 OR MORE CHARACTERS).  TEST PROGRAM
          C     PARAMETER TO SEE IF THIS WORD SHOULD BE SAVED.
      330            GO TO (23,25,23,25), SW1
          C SAVE WORD IN SUPL. RECORD 2
      331         23 SUPL2(SINDX2) = NWRDS  .
      332            DO 24 J = 1,NWRDS
      333            SINDX2=SINDX2+1
      334         24 SUPL2(SINDX2)=KWORD(J)
      336            SINDX2=SINDX2+1
      337            SUPL2(SINDX2)=0
          C ZERO WILL BE WRITTEN OVER BY NEXT ENTRY, OTHERWISE IT IS AN END OF
          C      RECORD SIGNAL.
          C ***********************************************
          C ****** WORD CHECK COMPLETED SECTION ******
          C ***********************************************
          C CHECK FOR A MULTIPLE WORD.
      340         25 IF (MULWRD.EQ.0) GO TO 27
          C IF THIS WORD IS VALID, GO TO VALID WORD SECTION.
      343            IF (IVALID.NE.0) GO TO 29
          C RESTORE SECOND (ADDENDED WORD) OF MULTIPLE WORD FOR A SEPARATE CHECK.
      346            J=1
      347            DO 26 K=SVLNGH,LENGTH
      350            WORD(J)=WORD(K)
      351         26 J=J+1
      353            LENGTH=LENGTH+1-SVLNGH
      354            MULWRD=0
      355            SAVE=0
          C GO BACK TO PROCESS THIS WORD.
      356            GO TO 8
          C REINITIALIZE FOR A NEW TEXT WORD
      357         27 LENGTH=0
      360            SAVE=0
      361            MAX=0
          C CHECK TO SEE IF END OF SENTENCE WAS REACHED.  IF SO, TURN OFF
```

```
571                    BAXEND                    FORTRAN SOURCE LIST EDIT
      ISN       SOURCE STATEMENT

          C     -PREVIOUS VALID WORD IN SENTENCE- INDICATOR.
      362       IF (ENDSEN.EQ.1) ISENT=0
      365       ENDSEN=0
      366       GO TO 5
          C ********************************
          C ******VALID WORD SECTION ******
          C ********************************
      367    28 IVALID=M
          C EVEN THOUGH A WORD IS VALID, IT IS POSSIBLE FOR IT TO BE PART OF A
          C MULTIPLE WORD, SO TEST TO SEE IF THIS VALID WORD IS A SINGLE WORD OR
          C   A MULTIPLE WORD.
      370       IF (MULWRD.EQ.1) GO TO 29
          C GO PICK UP SECOND WORD FROM TEXT.
      373       GO TO 17
          C IF THERE HAS BEEN ANOTHER VALID WORD IN THE SAME SENTENCE, GO TO
          C VALID WORD ASSOCIATION SECTION.
      374    29 IF (ISENT.NE.0) GO TO 38
          C PICK UP PACKAGE NUMBER OF THIS WORD FROM DICTIONARY.
      377    30 ITEMP = IABS(DICT(IVALID-1))
          C TEST PACKAGE NUMBER TO SEE IF WORD IS MEDICAL OR NON-MEDICAL.
      400       IF (ITEMP.LT.200) GO TO 34
          C WORD IS NON-MEDICAL, TEST PROGRAM PARAMETER TO SEE IF IT SHOULD BE
          C      SAVED.
      403       GO TO (31,36), SW2
      404    31 IVALID = PACK1(ITEMP-200)
          C SEARCH NON-MEDICAL FILE FOR THE SAME WORD.
      405       DO 32 J = 1,NMINDX
      406       IF (IVALID.EQ.NMWORD(J,1)) GO TO 33
      411    32 CONTINUE
          C PUT NEW WORD AND THE COUNT IN THE FILE.
      413       NMWORD(NMINDX,1) = IVALID
      414       NMWORD(NMINDX,2) = 1
      415       NMINDX = NMINDX+1
          C SET UP A NEW END OF FILE INDICATOR.
      416       NMWORD(NMINDX,1) = 0
      417       GO TO 36
          C SINCE WORD IS ALREADY IN THE FILE, JUST INCREASE ITS COUNT.
      420    33 NMWORD(J,2) = NMWORD(J,2)+1
      421       GO TO 36
          C PROCESS THE VALID MEDICAL WORD, SEARCH THE FILE FOR THE SAME WORD.
      422    34 DO 35 J=1,GINDEX
      423       IF (IVALID.EQ.GOODWD(J,1)) GO TO 37
      426    35 CONTINUE
          C PUT NEW WORD INTO LIST OF VALID MEDICAL WORDS.
      430       GOODWD(GINDEX,1)=IVALID
      431       GOODWD(GINDEX,2)=1
      432       GINDEX=GINDEX+1
          C THE ZERO AT THE END OF THE LIST IS AN END OF LIST SIGNAL
      433       GOODWD(GINDEX,1)=0
          C SET INDICATOR SHOWING A VALID WORD IN SENTENCE BEING PROCESSED.
      434    36 ISENT=IVALID
      435       IVALID=0
          C GO PICK UP A NEW TEXT WORD
      436       GO TO 25
          C SINCE WORD IS ALREADY IN LIST, ONLY INCREASE ITS COUNT.
```

```
571                    BAXEND                    FORTRAN SOURCE LIST EDIT
      ISN        SOURCE STATEMENT

      437     37 GOODWD(J,2)=GOODWD(J,2)+1
      440        GO TO 36
          C ***********************************************
          C ****** VALID WORD ASSOCIATIONS SECTION ******
          C ***********************************************
          C EXAMINE THE PRESENT WORD TO SEE IF IT IS MEDICAL OR NON-MEDICAL.
      441     38 TYPE1 = (IABS(DICT(IVALID-1))+99)/100
      442        GO TO (39,39,43), TYPE1
          C THE PRESENT WORD IS A MEDICAL TERM.
      443     39 TYPE2 = (IABS(DICT(ISENT-1))+99)/100
          C TEST THE PREVIOUS VALID WORD IN SENTENCE TO SEE IF IT IS A NONMEDICAL
          C     WORD.
      444        IF (TYPE2.NE.3) GO TO 30
      447        IVAL1=IVALID
      450        IVAL2=ISENT
          C CONVERT THE NON-MEDICAL WORD TO ITS BASIC TERM.
      451     40 IVAL2 = IABS(DICT(IVAL2-1))-200
      452        IVAL2 = PACK1(IVAL2)
      453        PACKNO = IABS(DICT(IVAL1-1))
          C CONVERT, IF NECESSARY, THE MEDICAL WORD TO ITS BASIC TERM.
      454        IF (PACKNO.EQ.50.OR. PACKNO .EQ.49)IVAL1=PACK4(4*PACKNO-2)
          C SEARCH THE LIST OF ASSOCIATED VALID WORDS FOR THE SAME TWO WORDS.
      457        DO 41 J=1,SINDX3
      460        IF (IVAL1.EQ.SUPL3(J,1).AND.IVAL2.EQ.SUPL3(J,2)) GO TO 42
      463     41 CONTINUE
          C PUT THE TWO NEW TERMS AT THE END OF THE LIST.
      465        SUPL3(SINDX3,1)=IVAL1
      466        SUPL3(SINDX3,2)=IVAL2
          C PUT IN A COUNT OF THE ASSOCIATION.
      467        SUPL3(SINDX3,3)=1
      470        SINDX3=SINDX3+1
          C SET UP AN -END OF LIST- INDICATOR.
      471        SUPL3(SINDX3,1)=0
      472        SUPL3(SINDX3,2)=0
      473        GO TO 30
          C THE TWO TERMS ARE ALREADY IN THE LIST, INCREASE THE COUNT.
      474     42 SUPL3(J,3) = SUPL3(J,3)+1
      475        GO TO 30
          C THE PRESENT VALID WORD IS NON-MEDICAL, TEST THE PREVIOUS VALID WORD
          C IN SENTENCE TO SEE IF IT IS A MEDICAL TERM.
      476     43 TYPE2=TYPE1
      477        TYPE1=(IABS(DICT(ISENT-1))+99)/100+1
      500        IF (TYPE1.NE.1.AND.TYPE1.NE.2) GO TO 30
      503        IVAL1=ISENT
      504        IVAL2=IVALID
      505        GO TO 40
          C ***************************************
          C ****** END OF ARTICLE SECTION ******
          C ***************************************
      506     44 GINDEX = GINDEX-1
          C PRINT THE NUMBER OF WORDS IN THIS ARTICLE.
      507        CALL CONCOD(WRDCT)
      510        IF(WDCT.LT.1000) GO TO 107
      513        RDCT = OR(RDCT,DOUBL)
      514        GO TO 108
```

```
571                    BAXEND                    FORTRAN SOURCE LIST EDIT
      ISN       SOURCE STATEMENT

      515    107 RDCT = OR(RDCT,TREBL)
      516    108 WRITE(6,45) WDCT, NOWRD
      517     45 FORMAT(1HO,I5,4A6,/)
      520        WRITE(4) NOLINE
      521        WRITE (4) WRDCT,NOWRD,(NOLINE(I),I=1,7)
      526        WRITE(4) NOLINE
          C      **********************************
          C      * PRINT VALID WORDS SUB-SECTION *
          C      **********************************
          C PROCESS LIST OF VALID WORDS.
      527        DO 60 I = 1,GINDEX
      530        KT = 0
          C TEST TO SEE IF THIS WORD WAS ALREADY DONE.
      531        IF (GOODWD(I,1).EQ.0) GO TO 60
      534        IVALID = GOODWD(I,1)
          C INITALIZE PRINT LINE INDEX.
      535        LINDEX = 0
      536        PACKNO = IABS(DICT(IVALID-1))
          C LOCATE THE PACKAGE FOR THIS WORD.
      537        PINDEX = PACKNO*4-3
          C GET THE REGISTRY NUMBER OUT OF THE PACKAGE.
      540        REGNO = PACK4(PINDEX)
          C      -*-*-*-*-*-*-*-*-*
      541        NP = NP+1
      542        IF (REGNO.EQ.0) PAKNOS(NP) = PACKNO
          C TEST FOR A NULL REGISTRY NUMBER OR ONE WITH AN MX PREFIX.
      545        IF (REGNO) 47,51,46
      546     46 ITEMP = REGNO
      547        GO TO 48
      550     47 ITEMP = -(REGNO+8000000)
          C PLACE MX-8 PREFIX IN LINE.
      551        LINE(1) = MA
      552        LINE(2) = X
      553        LINE(3) = DASH
      554        LINE(4) = EIGHT
      555        LINDEX = 4
          C CONVERT THE REGISTRY NUMBER (IN BINARY) TO CODED (A6 FORMAT).
      556     48 KWORD(1) = ITEMP
      557        KT = 1
      560        CALL CONCOD(KWORD(1))
          C CONVERT THE A6 FORMAT TO A1.
      561        NWRDS = 1
      562        CALL DECOMP
          C LOOK FOR A LEADING ZERO IN THE REGISTRY NUMBER.
      563        IF (ITEMP.LT.100000.AND.REGNO.GT.0) GO TO 49
      566        LINDEX = LINDEX+1
          C PUT THE FIRST DIGIT IN THE PRINT LINE.
      567        LINE(LINDEX) = WORD(1)
      570     49 DO 50 J = 2,6
          C PLACE LAST 5 DIGITS OF REGISTRY NUMBER IN PRINT LINE.
      571        LINDEX = LINDEX+1
      572     50 LINE(LINDEX) = WORD(J)
      574        LINDEX = LINDEX+1
      575        LINE(LINDEX) = BLANK
          C      -*-*-*-*-*-*-*-*-*
```

571 BAXEND FORTRAN SOURCE LIST EDIT

```
ISN       SOURCE STATEMENT

576         NP = NP-1
577         NR = NR+1
600         REGNOS(NR) = REGNO
      C INITIALIZE COUNT OF REFERENCES TO THIS PACKAGE.
601       51 PCOUNT = 0
602         KT = KT + 1
      C LOOK THROUGH THE REST OF THE LIST FOR A REFERENCE TO THE SAME PACKAGE,
      C     IF FOUND, INCREASE COUNT.
603         DO 52 J = I,GINDEX
604         IF (GOODWD(J,1).EQ.0) GO TO 52
607         ITEMP = GOODWD(J,1)
610         IF (IABS(DICT(ITEMP-1)).EQ.PACKNO) PCOUNT = PCOUNT+GOODWD(J,2)
613       52 CONTINUE
      C RECORD THE NUMBER OF REFERENCES IN PRINT LINE.
615         LINDEX = LINDEX+1
616         LINE(LINDEX) = LP
617         LINDEX = LINDEX+1
      C REDUCE THE PACKAGE REFERENCE COUNT TO AN INDICATOR.
620         IN = 2
621         IF (PCOUNT*1000.LE.WDCT) IN = 1
624         IF (PCOUNT*1000.GE.3*WDCT) IN = 3
627         GO TO (91,92),KT
630       91 PKNOWT(NP) = IN
631         GO TO 93
632       92 REGNWT(NR) = IN
633       93 KT = 0
634       53 LINE(LINDEX) = INDIC(IN)
635         LINDEX = LINDEX+1
636         LINE(LINDEX) = RP
637         LINDEX = LINDEX+1
640         LINE(LINDEX) = BLANK
      C NOW PROCESS EACH TERM CONTAINED IN THE PACKAGE.
641         DO 54 J = 1,3
642         JJ = PINDEX+J
643         PTERM = PACK4(JJ)
      C CHECK FOR A NULL TERM IN THE PACKAGE.
644         IF (PTERM.EQ.0) GO TO 54
      C PLACE THE PACKAGE TERM IN THE PRINT LINE.
647         CALL WRDPUT(PTERM,2)
650         LINDEX = LINDEX+1
651         LINE(LINDEX) = COMMA
652         LINDEX = LINDEX+1
653         LINE(LINDEX) = BLANK
      C EXAMINE THE REST OF THE VALID WORD LIST FOR THIS SAME PACKAGE TERM.
654         DO 54 K = I,GINDEX
655         IF (GOODWD(K,1).NE.PTERM) GO TO 54
      C STRIKE THIS WORD FROM THE LIST SINCE IT HAS BEEN PRINTED ALREADY.
660         GOODWD(K,1) = 0
661       54 CONTINUE
      C NOW EXAMINE THE LIST OF VALID WORDS FOR A WORD THAT BELONGS TO THE
      C     PACKAGE BEING PROCESSED BUT IS NOT CONTAINED IN THE PACKAGE.
664       55 DO 57 J= I,GINDEX
665         IF (GOODWD(J,1).EQ.0)  GO TO 57
670         ITEST = GOODWD(J,1)
671         IF (IABS(DICT(ITEST-1)).NE.PACKNO) GO TO 57
```

```
571                    BAXEND                    FORTRAN SOURCE LIST EDIT
      ISN          SOURCE STATEMENT

          C WE HAVE FOUND A WORD BELONGING TO THE PACKAGE BEING PROCESSED.
      674       GOODWD(J,1) = 0
          C TEST THE PACKAGE NUMBER, IF NEGATIVE DO NOT PRINT THE WORD.
      675       IF (DICT(ITEST-1)) 57,57,56
          C PUT THE WORD IN THE PRINT LINE.
      676    56 CALL WRDPUT(ITEST,2)
      677       LINDEX = LINDEX+1
      700       LINE(LINDEX) = COMMA
      701       LINDEX = LINDEX+1
      702       LINE(LINDEX) = BLANK
          C     -*-*-*-*-*-*-*-*-*-*
      703       NTN = NTN+1
      704       TRNAMS(NTN) = ITEST
      705    57 CONTINUE
          C CHECK FOR AN EMPTY PRINT LINE.
      707       IF (LINDEX.EQ.0) GO TO 59
          C DO NOT PRINT LAST COMMA AND BLANK.
      712       LINDEX =LINDEX-2
          C PRINT THE LAST LINE FOR THIS PACKAGE.
      713       WRITE (6,58) (LINE(J),J=1,LINDEX)
      720    58 FORMAT (1H ,132A1)
      721       CALL TAPREC
          C SKIP A LINE BETWEEN PACKAGES.
      722    59 WRITE (6,58)
      723       WRITE (4) NOLINE
      724    60 CONTINUE
          C     ****************************************
          C     * PRINT WORD ASSOCIATIONS SUB-SECTION *
          C     ****************************************
          C SKIP TWO LINES FOR NEXT OUTPUT SECTION.
      726       WRITE (6,61)
      727    61 FORMAT (1HO)
      730       WRITE (4) NOLINE
      731       WRITE (4) NOLINE
      732       SINDX3=SINDX3-1
          C     -*-*-*-*-*-*-*-*-*-*
          C SAVE THIS FILE -AS IS-.
      733       NC = SINDX3
      734       DO 81 J = 1,2
      735       DO 81 I = 1,NC
      736    81 COMPAR(I,J) = SUPL3(I,J)
          C PROCESS LIST OF WORD ASSOCIATIONS.
      741       DO 64 I = 1,SINDX3
          C SKIP ANY WORD ALREADY DONE.
      742       IF (SUPL3(I,1).EQ.0) GO TO 64
      745       M1 = SUPL3(I,1)
          C INITIALIZE FOR A NEW PRINT LINE.
      746       LINDEX = 0
          C PUT THE MEDICAL TERM IN THE PRINT LINE FOLLOWED BY A SLASH.
      747       CALL WRDPUT(M1,0)
      750       LINDEX = LENGTH+1
      751       LINE(LINDEX) = SLASH
          C SEARCH THE RECORD FOR THE SAME MEDICAL TERM.
      752       DO 63 J = I,SINDX3
      753       IF (SUPL3(J,1).NE.M1) GO TO 63
```

```
571                    BAXEND                    FORTRAN SOURCE LIST EDIT
      ISN       SOURCE STATEMENT

          C REMOVE THE MEDICAL TERM FROM THE FILE.
      756         SUPL3(J,1) = 0
          C REMOVE THE NON-MEDICAL TERM.
      757         M2 = SUPL3(J,2)
          C REMOVE THE COUNT FOR THIS ASSOCIATION.
      760         NUMOCC = SUPL3(J,3)
      761         LINDEX = LINDEX +1
      762         LINE(LINDEX) = BLANK
          C PUT THE NON-MEDICAL TERM IN THE PRINT LINE.
      763         CALL WRDPUT(M2,5)
          C INSERT A FREQUENCY COUNT INDICATOR.
      764         LINDEX = LINDEX+1
      765         LINE(LINDEX) = BLANK
      766         LINDEX = LINDEX+1
      767         LINE(LINDEX) = LP
      770         LINDEX = LINDEX+1
      771         IN = 2
      772         IF (NUMOCC*1000.LE. WDCT) IN = 1
      775         IF (NUMOCC*1000.GE.3* WDCT) IN = 3
     1000         COMPWT(J) = IN
     1001      62 LINE(LINDEX) = INDIC(IN)
     1002         LINDEX = LINDEX+1
     1003         LINE(LINDEX) = RP
     1004         LINDEX = LINDEX+1
     1005         LINE(LINDEX) = COMMA
     1006      63 CONTINUE
          C DO NOT PRINT LAST COMMA IN THE LINE.
     1010         LINDEX = LINDEX-1
          C PRINT THE INCOMPLETED LINE.
     1011         WRITE (6,58) (LINE(J),J=1,LINDEX)
     1016         CALL TAPREC
     1017      64 CONTINUE
          C       -*-*-*-*-*-*-*-*-*-*
     1021         WRITE (3) ARTNUM, NA,AUTHOR,NTN,TRNAMS,NR,REGNOS, REGNWT, NP,
                 1PAKNOS,PKNOWT,NC,COMPAR,COMPWT
          C       ****************************************
          C       * PRINT NON-MEDICAL WORDS SUB-SECTION *
          C       ****************************************
          C SKIP TWO LINES BEFORE DOING NEXT PRINT SECTION.
     1022         WRITE (6,61)
          C TEST PROGRAM PARAMETER TO SEE IF NON-MEDICAL WORDS ARE TO BE LISTED.
     1023         GO TO (65,68) , SW2
     1024      65 NMINDX = NMINDX-1
          C CHECK FOR AN EMPTY FILE.
     1025         IF (NMINDX.EQ.0) GO TO 68
          C INITIALIZE PRINT LINE INDEX.
     1030         LINDEX = 0
          C PROCESS THE NON-MEDICAL FILE.
     1031         DO 67 I = 1,NMINDX
          C TAKE THE WORD AND FREQUENCY COUNT FROM THE FILE.
     1032         M = NMWORD(I,1)
     1033         NUMOCC = NMWORD(I,2)
          C PUT NON-MEDICAL TERM IN PRINT LINE.
     1034         CALL WRDPUT(M,6)
          C PUT THE NUMBER OF OCCURANCE OF TERMS IN LINE.
```

571 BAXEND FORTRAN SOURCE LIST EDIT

```
 ISN       SOURCE STATEMENT

1035        LINDEX = LINDEX+1
1036        LINE(LINDEX) = LP
1037        NDIGIT = 1
1040        IF (NUMOCC.GT.9) NDIGIT = 2
      C CONVERT COUNT FROM BINARY TO CODED (A6 FORMAT).
1043        CALL CONCOD(NUMOCC)
      C CONVERT FROM A6 TO A1 FORMAT.
1044        KWORD(1) = NUMOCC
1045        NWRDS =1
1046        CALL DECOMP
1047        DO 66 J =1,NDIGIT
1050        JJ = 6+J-NDIGIT
1051        LINDEX = LINDEX+1
1052     66 LINE(LINDEX) = WORD(JJ)
1054        LINDEX = LINDEX+1
1055        LINE(LINDEX) = RP
1056        LINDEX = LINDEX+1
1057        LINE(LINDEX) = COMMA
1060        LINDEX = LINDEX+1
1061        LINE(LINDEX) = BLANK
1062     67 CONTINUE
1064        IF (LINDEX.EQ.0) GO TO 68
      C DO NOT PRINT THE LAST COMMA AND BLANK.
1067        LINDEX = LINDEX-2
      C PRINT THE LAST INCOMPLETE LINE.
1070        WRITE (6,58) (LINE(I),I=1,LINDEX)
      C     ***********************************
      C     * PRINT COMPARISONS SUB-SECTION *
      C     ***********************************
      C TEST PROGRAM PARAMETER TO SEE IF COMPARISONS ARE TO BE LISTED.
1075     68 GO TO (69,69,74,78) , SW1
1076     69 SVINDX = 1
      C PRINT HEADING.
1077        WRITE (6,70)
1100     70 FORMAT (1HO,15X,21HC O M P A R I S O N S/1H ,5X,6HDEGREE,5X,30HTEX
           1T WORD  -   DICTIONARY WORD)
      C CHECK FOR AN EMPTY FILE.
1101        IF (SUPL(1).NE.0) GO TO 72
1104        WRITE (6,71)
1105     71 FORMAT (1H ,15X,7HN O N E)
1106        GO TO 74
      C LOOK FOR THE -END OF FILE- INDICATOR.
1107     72 IF (SUPL(SVINDX).EQ.0) GO TO 74
1112        M=SUPL(SVINDX)
1113        MAX=SUPL(SVINDX+1)
1114        NWRDS=SUPL(SVINDX+2)
1115        N=(DICT(M-2))-3
1116        NWINDX=SVINDX+NWRDS+2
1117        JSV=SVINDX+3
      C PRINT THE TEXT WORD AND DICTIONARY WORD SHOWING DEGREE OF COMPARISON.
1120        WRITE (6,73) MAX,(SUPL(J),J=JSV,NWINDX),DASH,(DICT(K),K=M,N)
1131     73 FORMAT (1H ,7X,I2,3X,29A6)
1132        SVINDX=NWINDX+1
1133        GO TO 72
      C     ***********************************
```

571 BAXEND FORTRAN SOURCE LIST EDIT

```
 ISN       SOURCE STATEMENT

      C      * PRINT LARGE WORDS SUB-SECTION *
      C      **********************************
      C TEST PROGRAM PARAMETER TO SEE IF LARGE WORDS ARE TO BE LISTED.
1134    74 IF (SW1.EQ.2) GO TO 78
      C PRINT HEADING.
1137       WRITE (6,75)
1140    75 FORMAT (1HO,15X,21HL A R G E   W O R D S)
1141       SINDX2=1
      C CHECK FOR AN EMPTY FILE.
1142       IF (SUPL2(1).NE.0) GO TO 76
1145       WRITE (6,71)
1146       GO TO 78
      C LOOK FOR THE -END OF FILE- INDICATOR.
1147    76 IF (SUPL2(SINDX2).EQ.0) GO TO 78
1152       NWRDS=SUPL2(SINDX2)
1153       NWINDX=SINDX2+NWRDS
1154       JSIN=SINDX2+1
      C PRINT THE LARGE WORD.
1155       WRITE (6,77) (SUPL2(J),J=JSIN,NWINDX)
1162    77 FORMAT (1H ,17X,14A6)
1163       SINDX2=NWINDX+1
1164       GO TO 76
      C IDENTIFY THE END OF THE ARTICLE
1165    78 WRITE (6,79) EOA
1166    79 FORMAT (1HO,4A6)
      C     -*-*-*-*-*-*-*-*-*-*
1167       WRITE (4) NOLINE
1170       WRITE (4) EOA,(NOLINE(I),I=1,8)
1175       ARTNUM = ARTNUM+1
      C GO CHECK FOR A NEW ARTICLE
1176       GO TO 2
1177    80 FORMAT (14A6)
1200       END
```

571 BAXEND FORTRAN SOURCE LIST

```
ISN       SOURCE STATEMENT

  0 $IBFTC HEAD
  1       SUBROUTINE HEAD
    C     THIS SUBROUTINE READS IN THE BIBLOGRAPHIC ENTRY CARDS OF EACH
    C     ARTICLE.  IT CONTINUES READING CARDS UNTIL A SIGNAL CARD IS
    C     ENCOUNTERED.  THE INFORMATION ON THESE CARDS ARE PRINTED WITH
    C     A MARGIN WIDTH THAT IS SPECIFIED IN THE LABELLED COMMON AREA -H-
  2       INTEGER V,Z,U,Y,W
  3       INTEGER RECORD(72),PRINT(132),EX,BLANK
  4       INTEGER WORD,AUTHOR,COMMA,ARTNUM
  5       COMMON /H/ NUM,ARTNUM,NAUTH,AUTHOR(6,5)
  6       COMMON /LIN/ PRINT,LINDEX,LM,IX(3000)
  7       COMMON KWORD(14),NWRDS,WORD(72),LENGTH,IEND
 10       DATA EX,BLANK/1HX,1H /
 11       DATA COMMA/1H,/
 12       NAUTH = 1
 13       LENGTH = 30
 14       READ(15,4) (RECORD(Z),Z=1,72)
 21       IF(RECORD(1).EQ.EX.AND.RECORD(2).EQ.EX.AND.RECORD(3).EQ.EX)
         1GO TO 30
 24       IEND=1
 25       GO TO 31
 26    30 IEND=2
 27       RETURN
 30    31 J = 1
 31       WRITE(6,32) ARTNUM
 32    32 FORMAT(1H1,14HARTICLE NUMBER,I5)
 33       WRITE (4) ARTNUM
 34    20 DO 21 I = 1,30
 35    21 WORD(I) = BLANK
 37       I = 1
 40    22 IF (RECORD(J).EQ.BLANK) GO TO 26
 43    23 IF (RECORD(J).EQ.COMMA) GO TO 24
 46       WORD (I) = RECORD(J)
 47       I = I + 1
 50       J = J + 1
 51       GO TO 22
 52    24 CALL COMPOS
 53       DO 25 I = 1,5
 54    25 AUTHOR(NAUTH,I) = KWORD(I)
 56       NAUTH = NAUTH + 1
 57       J = J + 2
 60       GO TO 20
 61    26 IF (RECORD(J+1).NE.BLANK) GO TO 23
 64       CALL COMPOS
 65       DO 27 I = 1,5
 66    27 AUTHOR(NAUTH,I) = KWORD(I)
 70       J = 1
 71       W = 1
 72       M = 0
 73       GO TO 28
    C     SET UP OUTPUT PARAMETER J.
 74     2 J = 1
 75       W = 1
    C     SET UP INPUT PARAMETER M.
 76     3 M = 0
```

571 BAXEND FORTRAN SOURCE LIST HEAD

```
ISN       SOURCE STATEMENT

 77         READ(15,4)  (RECORD(Z), Z = 1,72)
104       4 FORMAT(72A1)
105      28 IF(.NOT.((RECORD(1).EQ.BLANK).AND.(RECORD(2).EQ.BLANK))) GO TO 6
110         DO 5  Y = W,NUM
111         PRINT(Y) = BLANK
112       5 CONTINUE
114         GO TO 12
      C     CHECK FOR THE SIGNAL CARD.
115       6 IF (RECORD(1) .EQ. EX) RETURN
120       7 DO 9  I = J,NUM
121         M = M+1
122         PRINT(I) = RECORD(M)
123         IF (M .LT. 72) GO TO 8
126         IF (I.EQ.NUM) GO TO 12
131         GO TO 11
132       8 IF ((RECORD(M).EQ.BLANK).AND.(RECORD(M+1).EQ.BLANK).AND.(RECORD(M+
           12).EQ.BLANK))  GO TO 12
135       9 CONTINUE
      C     IF WORD IS INCOMPLETE AT END OF OUTPUT LINE, MOVE ENTIRE WORD
      C        TO NEXT LINE.
137         L = NUM
140      10 IF (PRINT(L) .EQ. BLANK) GO TO 13
143         L = L-1
144         GO TO 10
145      11 J = I+1
146         GO TO 3
147      12 WRITE (6,14) (PRINT(K), K = 1,I)
154         WRITE (7,19) (PRINT(K),K=1,I)
161         LINDEX = I
162          CALL TAPREC
163         GO TO 2
164      13 WRITE (6,14) (PRINT(K), K = 1,L)
171      14 FORMAT (1H ,130A1)
172         WRITE (7,19) (PRINT(K),K=1,L)
177         LINDEX = L
200         CALL TAPREC
201         IF (L.LT.NUM) GO TO 15
204         J = 1
205         W = 1
206         GO TO 17
207      15 V = 0
210         L = L+1
211         DO 16 N = L,NUM
212         V = V+1
213         PRINT(V) = PRINT(N)
214      16 CONTINUE
216         J = V+1
217         W = J
220      17 IF (M .EQ. 72) GO TO 3
223         GO TO 7
224      19 FORMAT (132A1)
225         END
```

5. Revised Version of Dictionary Processing Program

31571 BAXEND FORTRAN SOURCE LIST

```
ISN       SOURCE STATEMENT

  0 $IBFTC DICTDO
    C      DICTIONARY PROCESSING AND CREATION OF TRANSFER MATRIX
  1        COMMON INDICT(12),DICT(3000),TRADD(4096)
  2        INTEGER DICT,TRADD,END,BLANK
  3        DATA END,BLANK/6HENDICT,6H      /
  4        WRITE (6,71)
  5     71 FORMAT  (33H1  ADDR PACK NO   DICTIONARY TERM/)
    C      LINK IS A VALUE USED TO CHAIN DICTIONARY ENTRIES
    C      NLINK IS USED TO GENERATE NEXT LINK VALUE
    C      TWO WORDS PRECEDE EACH DICTIONARY TERM - LINKAGE AND PACKAGE
    C      NUMBER - HENCE FIRST TERM STARTS IN LOCATION 3
  6        LINK = 3
  7        NLINK=3
    C      INDX REFERS TO INDEX OF TRANSFER MATRIX
 10        INDX=0
 11        LB = 1
    C      READ IN ONE LOGICAL RECORD - ONE DICTIONARY ENTRY
 12      1 READ(5,2) INDICT,NPACK
 15      2 FORMAT (12A6,4X,I4)
 16        LBINK = LINK
 17        LINK = NLINK
    C      CHECK FOR END OF DICTIONARY CARDS
 20        IF (INDICT(1).EQ.END) GO TO 10
    C      NWRD REFERS TO NUMBER OF MACHINE WORDS USED BY DICTIONARY ENTRY
 23        DO 3 NWRD =1,12
 24        IF (INDICT(NWRD).EQ.BLANK) GO TO 5
 27        DICT(NLINK)=INDICT(NWRD)
 30      3 NLINK=NLINK+1
    C      IF PROGRAM COMES TO THIS POINT AN ERROR EXISTS
 32        WRITE(6,4) INDICT
 33      4 FORMAT(1H ,21HERROR 1, LARGE ENTRY., 14A6)
 34        CALL EXIT
 35      5 NLINK=NLINK+2
 36        L = NWRD - 1
    C      STORE LENGTH OF TERM IN PACKAGE NUMBER WORD
 37        IF(NPACK) 32,31,31
 40     31 NPACK = NPACK + L*2**15
 41        GO TO 30
 42     32 NPACK = -(IABS(NPACK) + L*2**15)
 43     30 DICT(LINK-1)=NPACK
 44        DICT(LINK-2)=NLINK
 45        WRITE (6,70) NPACK,NLINK,INDICT
 46     70 FORMAT (1H ,2I10,5X,12A6)
 47     33 IF (INDICT(1).LT.0) GO TO 6
    C      MXINDX IS THE MATRIX INDEX GENERATED BY A DICTIONARY ENTRY
 52        MXINDX=INDICT(1)/2**24
 53        GO TO 7
 54      6 MXINDX=2**11-INDICT(1)/2**24
 55      7 IF (MXINDX-INDX) 1,9,8
 56      8 INDX=INDX+1
 57        TRADD(INDX)=0
 60        GO TO 7
 61      9 INDX=INDX+1
 62        TRADD(INDX)=LINK
 63        GO TO (13,15),LB
```

31571 BAXEND FORTRAN SOURCE LIST DICTDO

```
ISN       SOURCE STATEMENT

 64     13 LB = 2
 65        GO TO 1
 66     15 DICT(LBINK-2) = 0
 67        LB=2
 70        GO TO 1
    C      SET CHAIN WORD OF LAST ENTRY TO 0 AND LAST TERM TO END
 71     10 DICT(LINK-1)=0
 72        DICT(LINK)=INDICT(1)
    C      AT THIS POINT THE DICTIONARY IS (LINK) WORDS LONG
 73        WRITE(6,11) TRADD
 74     11 FORMAT(1HO,10I6)
 75        WRITE(6,72) LINK,(DICT(I),I = 1,LINK)
102     72 FORMAT(1HO,I8//(1H ,20A6))
103        WRITE (2) TRADD
104        WRITE (2) LINK,(DICT(I),I=1,LINK)
    C      BEGIN PROCESSING PACKAGE CARDS
111        INTEGER PACKNO,REGNO,PACK1(100),PACK5(350),PINDEX
112        DIMENSION INPUT(12)
113     40 READ (5,41) PACKNO,REGNO
116     41 FORMAT (I6,I8)
117        IF (PACKNO.EQ.999999) GO TO 67
122        I=1+PACKNO/100
123        GO TO (42,42,56), I
    C ************************************
    C ****  4-WORD PACKAGE PROCESSING  ****
    C ************************************
    C      COMPUTE THE INDEX FOR THE 4-WORD PACKAGE
124     42 PINDEX=PACKNO*4-3
125        PACK5(PINDEX)=REGNO
    C READ AND PROCESS 3 DATA CARDS
126        DO 52 I=1,3
127        PINDEX=PINDEX+1
130        READ (5,43) INPUT
132     43 FORMAT (12A6)
133        IF (INPUT(1).EQ.BLANK) GO TO 52
136        DO 44 NWRDS =1,12
137        IF (INPUT(NWRDS).EQ.BLANK) GO TO 45
142     44 CONTINUE
144     45 IF (INPUT(1).LT.0) GO TO 46
    C      INDEX REFERS TO INDEX OF TRANSFER ADDRESS MATRIX
147        INDEX=1+INPUT(1)/2**24
150        GO TO 47
151     46 INDEX=1+2**11-INPUT(1)/2**24
    C      USE M AS INDEX OF PROCESSED DICTIONARY
152     47 M=TRADD(INDEX)
153        NWRDS=NWRDS-1
    C      EXTRACT LENGTH OF DICTIONARY TERM FROM PACKAGE WORD - DICT(M-1)
154        K = 1
155     81 LENGTH = IABS(DICT(M-1))/2**15
156        IF(LENGTH.NE.NWRDS) GO TO (80,82),K
161        MTEMP = M
162        DO 48 L = 1,NWRDS
163        IF(INPUT(L).NE.DICT(MTEMP)) GO TO 80
166     48 MTEMP = MTEMP + 1
    C INSERT DICTIONARY INDEX IN PACKAGE
```

```
ISN       SOURCE STATEMENT

170          PACK5(PINDEX) = M
171          GO TO 52
         C   PACKAGE TERM DOES NOT MATCH DICTIONARY TERM GO TO NEXT TERM IN
         C   DICTIONARY
172       80 M = DICT(M-2)
173          IF(M.EQ.0) K= 2
176          GO TO 81
177       82 WRITE(6,50) PACKNO, INPUT
200       50 FORMAT(12HORE PACK NO., I4, 13H CAN NOT FIND/1H ,5X,12A6, 13HIN DI
             1CTIONARY)
201       52 CONTINUE
         C   READ IN NEXT PACKAGE DATA CARD
203          GO TO 40
         C ***********************************
         C *** 1 WORD PACKAGE PROCESSING ***
         C ***********************************
204       56 PINDEX=PACKNO-200
205          READ (5,43) INPUT
207          DO 58 NWRDS=1,12
210          IF (INPUT(NWRDS).EQ.BLANK) GO TO 59
213       58 CONTINUE
215       59 NWRDS=NWRDS-1
216          IF (INPUT(1).LT.0) GO TO 60
221          INDEX=1+INPUT(1)/2**24
222          GO TO 61
223       60 INDEX=1+2**11-INPUT(1)/2**24
224       61 M=TRADD(INDEX)
225          K = 1
226       63 LENGTH = IABS(DICT(M-1))/2**15
227          IF(LENGTH.NE.NWRDS) GO TO (77,79),K
232          MTEMP = M
233          DO 62 L = 1, NWRDS
234          IF(INPUT(L).NE.DICT(MTEMP)) GO TO 77
237       62 MTEMP = MTEMP + 1
241          PACK1(PINDEX) = M
242          GO TO 40
243       77 M = DICT(M-2)
244          IF(M.EQ.0) K = 2
247          GO TO 63
250       79 WRITE(6,50) PACKNO, INPUT
251          GO TO 40
252       67 WRITE(6,74)
253       74 FORMAT (22H1CODED TYPE 1 PACKAGES/)
254          WRITE (2) PACK5,PACK1
255          WRITE (6,68) (PACK5(I),I=1,236)
262       68 FORMAT (1HO,I9,3I5)
263          WRITE (6,73)
264       73 FORMAT (22H1CODED TYPE 2 PACKAGES/)
265          WRITE (6,69) (I,PACK1(I),I=1,15)
272       69 FORMAT (1HO,I9,I5)
273          CALL UPDAT
274          CALL EXIT
275          END
```

31571 BAXEND FORTRAN SOURCE LIST

```
ISN       SOURCE STATEMENT

  0 $IBFTC UPDAT
  1       SUBROUTINE UPDAT
    C     ROUTINE FOR UPDATING DICTIONARY TAPE
  2       INTEGER END,BLANK, DICT, TRADD
  3       COMMON NEWDCT(12), DICT(3000), TRADD(4096)
  4       DATA END,BLANK/6HENDICT,6H      /
    C     FIND END OF DICTIONARY OR LOCATION WHERE NEW DICTIONARY TERM MAY
    C     BE PLACED
  5       DO 1 I = 1,3000
  6       IF(DICT(I).EQ.END) GO TO 2
 11     1 CONTINUE
 13     2 MT = I
 14       M = MT
 15     3 READ(5,4) NEWDCT, NPACK
 20     4 FORMAT(12A6, 4X, I4)
 21       ISW = 1
 22       M = MT
 23       IF(NEWDCT(1) .EQ.END) GO TO 16
 26     5 IF(NEWDCT(1).LT. 0) GO TO 6
 31       MXINDX = NEWDCT(1)/2**24 + 1
 32       GO TO 7
 33     6 MXINDX = 2**11 - NEWDCT(1)/2**24 + 1
 34     7 IF(TRADD(MXINDX).EQ. 0) GO TO 17
 37       N = TRADD(MXINDX)
    C     FIND END OF STRING OF THIS GROUP OF TERMS
 40     8 IF(DICT(N-2).EQ.0) GO TO 9
 43       N = DICT(N-2)
 44       GO TO 8
 45     9 DO 10 NWRD = 1,12
 46       IF(NEWDCT(NWRD).EQ.BLANK) GO TO 12
 51       DICT(MT) = NEWDCT(NWRD)
 52    10 MT = MT + 1
 54       WRITE(6,11) NEWDCT
 55    11 FORMAT(1H ,21HERROR 1, LARGE ENTRY., 14A6)
 56    12 LENGTH = NWRD - 1
 57       MT = MT + 2
 60       IF(ISW.EQ.1) DICT(N-2) = M
    C     INSERTION OF LENGTH IN WORD CONTAINING PACKAGE NUMBER
 63       IF(NPACK) 13,14,14
 64    14 NPACK = NPACK + LENGTH*2**15
 65       GO TO 15
 66    13 NPACK = -(IABS(NPACK) + LENGTH*2**15)
 67    15 DICT(M - 1) = NPACK
 70       GO TO 3
    C     PLACE ADDRESS OF NEW TERM IN MATRIX
 71    17 TRADD(MXINDX) = M
 72       ISW = 2
 73       GO TO 9
    C     PUT END SENTINEL IN LOCATION FOR NEW WORD IN NEXT UPDATE RUN
 74    16 DICT(M) = END
 75       WRITE (6,18) M,(DICT(I),I=1,M)
102    18 FORMAT(1HO,I8//(1H ,20A6))
103       WRITE(6,19) TRADD
104    19 FORMAT(1HO, 10I6)
105       RETURN
106       END.
```

6. Search Program Listing

```
ISN       SOURCE STATEMENT

  0 $IBFTC SEARCH
  1        DIMENSION KOMPAR(100,2)
  2        INTEGER BARBIT(15),HYDANT(14),OXAZOL(7),SUCCIN(4),TERM(4)
  3        DATA BARBIT/15,50066,50113,57307,57330,57432,64437,76948,115388,
          1125337,-8028679,-8028691,-8028704,-8028715,-8028726/
  4        DATA HYDANT/14,50124,57410,86351,125611,630933,830897,3784927,
          15696060,6509348,-8028679,-8028691,-8028704,-8028726/
  5        DATA OXAZOL/7,115673,127480,520774,526352,695534,4171113/
  6        DATA SUCCIN/4,77418,77678,86340/
  7        DATA TERM/6HBARBIT,6HHYDANT,6HOXAZOL,6HSUCCIN/
 10        DIMENSION INPUT(12),NSAT(20)
 11        DIMENSION IDENT(20,24,2),NGEN(20)
 12        DIMENSION LOGX(20,72)
 13        DIMENSION LOGEXP(72)
 14        DIMENSION MASK(6),AMSK(6),DIG(6),NDG(6)
 15        INTEGER   RNWT(30), QRN(30)
 16        INTEGER CHARST(26),BLANK, RP, CC, CC1, CC2, CC3,
          1EXV,QN,DNA,RO,PERIOD, QTV
 17        INTEGER TT, QAUTHR(20,5), TRADD(4096), DICT(3000),  SSW1, SSW2,
          1 PACK4(350), PACK1(100), PACKNO, PINDEX, QTRNMS(30), REGNO,
          2 QREGNS(30), QPAKNS(30),QCOMP(100,2), SATIS(20,35),AUTHOR(6,5),
          3 COMPAR(100,2), TRNAMS(30), PAKNOS(30), REGNOS(30), AN,
          4 COMPWT(100),PKNOWT(30),REGNWT(30),QWT,RQWT(30),PQWT(30),CQWT(100)
 20        EQUIVALENCE (DIG(1),NDG(1)),(MASK(1),AMSK(1)),(PUTIN,INPUT(2))
 21        DATA MASK/0770000000000,07700000000,077000000,0770000,07700,077/
 22        DATA IBLANK/6H      /
 23        DATA CHARST/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
          11HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/,MINUS/1H-/,
          2LP/1H(/,RP/1H)/,DNA/1H*/,RO/1H+/,BLANK/1H / ,PERIOD/1H./
 24        COMMON IS(500),LST
    C      INPUT
 25        REWIND 3
 26        READ (3) TRADD
 30        READ (3) J,(DICT(I),I=1,J)
 36        READ (3) PACK4,PACK1
 41    137 READ (3) NUMART,AN
 44        IF(AN.NE.999) GO TO 137
 47        NUMART=NUMART-1
 50        REWIND 3
 51        READ (3) TRADD
 53        READ (3) J,(DICT(I),I=1,J)
 61        READ (3) PACK4,PACK1
 64        WRITE(6,138)
 65    138 FORMAT(1H1,20X,21HLIST OF INPUT QUERIES,//1H ,8HQUERY  1)
    C      INITIALIZE QUERY NUMBER, COUNTS.
 66        IWRSW = 1
 67        NQ = 1
 70        NAQ = 0
 71        NTNQ = 0
 72        NRQ = 0
 73        NPQ = 0
 74        NCQ =  0
 75        NRR = 0
 76        SSW2 = 1
    C      BEGIN.  READ A QUERY TERM CARD.
```

```
ISN       SOURCE STATEMENT

    C DESCRIPTION OF QUERY DATA CARDS
    COLUMN 1 ... A LETTER REPRESENTING THE QUERY TERM.   THIS MUST NOT BE Y
    C OR Z WHICH ARE RESERVED FOR THE LOGIC VALUES TRUE OR FALSE.
    C THE SAME LETTER MAY NOT BE USED TWICE IN ONE QUERY.
    COLUMN 3 ... A NUMERIC DIGIT SIGNIFYING THE TYPE OF QUERY TERM.
    C THE FOLLOWING CODES ARE USED.
    COLUMNS 4 THROUGH 76 CONTAIN THE QUERY TERM
    COLUMN 80 ... A NUMERIC DIGIT 1, 2 OR 3 TO INDICATE FREQUENCY INDEX OR WEIGHT.
 77      1 READ (5,2) LETTER,TT,INPUT,QWT
104      2 FORMAT (A1,I2,12A6,4X,A1)
105        IF(IWRSW.EQ.2.AND.TT.NE.7) WRITE(6,700) NQ
110    700 FORMAT(1H ,6HQUERY ,I2)
111        IWRSW = 1
112        WRITE(6,139) LETTER,TT,INPUT,QWT
113    139 FORMAT (1H ,10X,A1,I2,12A6,A1)
114        QWT = QWT/2**30
115        IF (QWT.LT.1) QWT=1
    C THE FOLLOWING FORMAT IS USED ON THE QUERY CARDS TO SIGNIFY TERM TYPE
    C      1 = AUTHOR
    C      2 = MEDICAL TERM (CAN BE TRADE NAME OR GENERIC NAME).
    C      3 = REGISTRY NUMBER.
    C      4 = ASSOCIATIONS (MUST BE TWO CARDS TOGETHER)
    C      5 = GENERIC TERM
    C      6 = END OF QUERY
    C      7 = END OF ALL QUERIES
120        DO 46 LET = 1,26
121     46 IF (LETTER.EQ.CHARST(LET)) GO TO 47
125        IF (LETTER.EQ.BLANK) GO TO 47
130        GO TO 91
131     47 IERR = 1
132        SSW1 = 1
    C      TEST FOR TYPE OF TERM
133        GO TO (3,5,28,22,247,25,26), TT
    C      THIS QUERY TERM IS AN AUTHORS NAME.
134      3 NAQ = NAQ+1
135        IDENT(NQ,LET,1) = 1
136        IDENT(NQ,LET,2) = NAQ
137        DO 4 I = 1,5
140      4 QAUTHR(NAQ,I) = INPUT(I)
    C      GO TO NEXT QUERY TERM.
142        GO TO 1
    C      ------------------
    C      THIS QUERY TERM IS A MEDICAL TERM.
143      5 DO 6 NWRDS = 1,11
144      6 IF (INPUT(NWRDS+1).EQ.IBLANK) GO TO 7
150        NWRDS = 12
    C      SEARCH THE DICTIONARY FOR THE TERM.
151      7 IF (INPUT(1).LT.0) GO TO 8
154        INDEX = INPUT(1)/2**24+1
155        GO TO 9
156      8 INDEX = 1+2**11-INPUT(1)/2**24
157      9 M = TRADD(INDEX)
160     10 IF (M.EQ.0) GO TO 92
163        IF (INPUT(1)) 11,89,12
164     11 IF (INPUT(1).GT.DICT(M)) GO TO 92
```

```
ISN       SOURCE STATEMENT

167          GO TO 13
170       12 IF (DICT(M).GT.INPUT(1)) GO TO 92
173       13 MTEMP = M
      C      NOW CHECK DICTIONARY WORD FOR A FULL COMPARISON.
174          DO 14 I = 1,NWRDS
175          IF (DICT(MTEMP).NE.INPUT(I)) GO TO 15
200       14 MTEMP = MTEMP+1
      C      NOW TEST QUERY TERM AND DICTIONARY TERM FOR EQUAL LENGTHS.
202          IF (NWRDS.EQ.(DICT(M-2)-(M+2))) GO TO 16
      C      WE DO NOT HAVE A COMPLETE COMPARISON.  GET NEXT DICTIONARY INDEX.
205       15 M = DICT(M-2)
206          GO TO 10
      C      WE HAVE NOW LOCATED THE QUERY TERM IN THE DICTIONARY.
      C      TEST SSW1.
207       16 GO TO (17,23), SSW1
      C      PICK UP PACKAGE NUMBER.
210       17 PACKNO = DICT(M-1)
      C      TEST FOR SPECIAL PACKAGE NUMBERS.
211          IF (IABS(PACKNO).EQ.49) GO TO 69
214          IF (IABS(PACKNO).EQ.50) GO TO 69
      C      GENERATE PACKAGE INDEX.
217       69 PINDEX = IABS(PACKNO)*4-3
220          IF (PACKNO) 20,94,18
      C      SEARCH PACKAGE FOR TERM.
221       18 DO 19 I = 1,3
222          J = PINDEX +I
223       19 IF (M.EQ.PACK4(J)) GO TO 20
      C      WE HAVE A TRADE NAME.
227          NTNQ = NTNQ+1
230          QTRNMS(NTNQ)    = M
231          IDENT(NQ,LET,1) = 2
232          IDENT(NQ,LET,2) = NTNQ
233          GO TO 1
      C      PICK UP THE REGISTRY NUMBER.
234       20 REGNO = PACK4(PINDEX)
235          IF (REGNO.EQ.0) GO TO 21
240          NRQ = NRQ+1
      C      RECORD THE REGISTRY NUMBER.
241          QREGNS(NRQ)    = REGNO
242          RQWT(NRQ) = QWT
243          IDENT(NQ,LET,1) = 3
244          IDENT(NQ,LET,2) = NRQ
245          GO TO 1
246       21 NPQ = NPQ+1
      C      RECORD THE PACKAGE NUMBER.
247          QPAKNS(NPQ)    = IABS(PACKNO)
250          PQWT(NPQ) = QWT
251          IDENT(NQ,LET,1) = 4
252          IDENT (NQ,LET,2) = NPQ
253          GO TO 1
      C      -----------------
      C WE HAVE AN ASSOCIATION TERM. SET UP SSW1 FOR A RETURN FROM THE
      C DICTIONARY LOOK-UP
254       22 SSW1 = 2
255          GO TO 5
```

```
ISN       SOURCE STATEMENT

256     23 IF (SSW2.EQ.2) GO TO 24
261        NCQ = NCQ + 1
262        QCOMP(NCQ,1) = IABS(DICT(M-1))
263        CQWT(NCQ) = QWT
264        IDENT(NQ,LET,1) = 5
265        IDENT(NQ,LET,2) = NCQ
     C SET SSW2 FOR A RETURN WITH THE SECOND ASSOCIATION TERM
266        SSW2 = 2
267        GO TO 1
270     24 PACKNO = IABS(DICT(M-1))
     C     RESET SSW2.
271        SSW2 = 1
     C     CHECK PACKAGE NUMBER.
272        IF (PACKNO.LE.200) GO TO 90
     C     NORMALIZE THE TYPE2 TERM.
275        PINDEX = PACKNO-200
276        QCOMP(NCQ,2) = PACK1(PINDEX)
277        GO TO 1
     C THIS QUERY TERM IS A GENERIC TERM
300    247 IDENT(NQ,LET,1) = 7
301        NGEN(NQ) = INPUT(1)
302        GO TO 1
     C     ---------------
     C     WE HAVE REACHED THE END OF THIS QUERY.
     C NOW READ LOGIC STATEMENT.
303     25 READ (5,29) (LOGX(NQ,I),I=1,72)
310     29 FORMAT (72A1)
311        WRITE(6,140)(LOGX(NQ,I),I=1,72)
316    140 FORMAT(1H ,10X,72A1,//)
317        IWRSW = 2
320        NQ = NQ+1
     C     GO LOOK FOR THE NEXT QUERY.
321        GO TO 1
     C     ---------------
     C THIS QUERY TERM IS A REGISTRY NUMBER
     C     ROUTINE TO CONVERT A SIX DIGIT BCD REGISTRY NUMBER TO BINARY FORM
322     28 DO 129 I = 1,5
323        J = 36-6*I
324        DIG(I) = AND(PUTIN,AMSK(I))
325    129 NDG(I) = NDG(I)/2**J
327        DIG(6) = AND(PUTIN,AMSK(6))
330        NQR = NDG(6) + NDG(5)*10 +NDG(4)*100 + NDG(3)*1000 + NDG(2)*10000
          1 + NDG(1)*100000
331        NRR  = NRR + 1
332        QRN(NRR) = NQR
333        RNWT(NRR) = QWT
334        IDENT(NQ,LET,1) = 6
335        IDENT(NQ,LET,2) = NRR
336        GO TO 1
     C    RESET THE NUMBER OF QUERIES.
337     26 NQ = NQ-1
     C     INITIALIZE THE ARRAY OF SATISFIED QUERIES.
340        DO 27 I = 1,NQ
341        NSAT(I) = 0
342        DO 27 J = 1,35
```

```
ISN       SOURCE STATEMENT

343     27 SATIS(I,J) = 0
346     30 IF(AN.EQ.NUMART) GO TO 81
351        READ (3) AN,NA,AUTHOR,NTN,TRNAMS,NR,REGNOS,REGNWT,NP,PAKNOS,PKNOWT
          1,NC,COMPAR,COMPWT
370        DO 75 QN = 1,NQ
371        DO 48 J = 1,72
372     48 LOGEXP(J) = LOGX(QN,J)
374        CC = 0
375        IND = 1
      C ******************************
      C *********  EVAL ROUTINE  ***
      C ******************************
376     49 M= 1
377     50 CC = CC +1
400        IF (LOGEXP(CC).EQ.BLANK) GO TO 50
403        IF (LOGEXP(CC).NE.MINUS) GO TO 51
406        M = -M
407        GO TO 50
      C CHECK FOR ANY NEW EXPRESSION BEGINING WITHIN PARENTHESES.
410     51 IF (LOGEXP(CC).EQ.LP) GO TO 66
413        DO 52 I = 1,26
414     52 IF (LOGEXP(CC).EQ.CHARST(I)) GO TO 31
      C ERROR CHECK FOR NON-VALID CHARACTER.
420        GO TO 98
      C DO NOT SEARCH FOR ANY TERM IF Y (TRUE) OR Z (FALSE) IS IN LINE.
421     31 IF (I.EQ.25) GO TO 34
424        IF (I.EQ.26) GO TO 36
      C IDENTIFY AND SEARCH FOR TERM REPRESENTED BY CHARACTER.
427        IF(IDENT(QN,I,1).EQ.7) GO TO 147
432        TT = IDENT(QN,I,1)
433        INDEX = IDENT(QN,I,2)
434        IF (TT.LE.0.OR.TT.GE.7) GO TO 100
437        GO TO (32,37,39,41,43,239),TT
      C SEARCH LIST OF AUTHORS.
440     32 DO 35 I = 1,NA
441        DO 33 J = 1,5
442     33 IF (AUTHOR(I,J).NE.QAUTHR(INDEX,J)) GO TO 35
446     34 QTV = 1
447        GO TO 53
450     35 CONTINUE
452     36 QTV = -1
453        GO TO 53
      C SEARCH LIST OF TRADE NAMES.
454     37 IF (NTN.EQ.0) GO TO 36
457        DO 38 I = 1,NTN
460     38 IF (TRNAMS(I).EQ.QTRNMS(INDEX)) GO TO 34
464        GO TO 36
      C SEARCH LIST OF REGISTRY NUMBERS.
465     39 IF (NR.EQ.0) GO TO 36
470        DO 40 I = 1,NR
471     40 IF (REGNOS(I).EQ.QREGNS(INDEX)) GO TO 234
475        GO TO 36
476    234 IF (REGNWT(I).GE.RQWT(INDEX)) GO TO 34
501        GO TO 36
      C SEARCH LIST OF PACKAGE NUMBERS.
```

```
ISN       SOURCE STATEMENT

502     41 IF (NP.EQ.0) GO TO 36
505        DO 42 I = 1,NP
506     42 IF ( PAKNOS(I).EQ.QPAKNS(INDEX)) GO TO 134
512        GO TO 36
513    134 IF (PKNOWT(I).GE.PQWT(INDEX)) GO TO 34
516        GO TO 36
     C SEARCH LIST OF ASSOCIATIONS.
517     43 IF (NC.EQ.0) GO TO 36
522        DO 45 I = 1,NC
523        MM = COMPAR(I,1)
524        KOMPAR(I,1) = IABS(DICT(MM-1))
525        KOMPAR(I,2) = COMPAR(I,2)
526        DO 44 J = 1,2
527     44 IF (KOMPAR(I,J).NE.QCOMP(INDEX,J)) GO TO 45
533        IF (COMPWT(I).GE.CQWT(INDEX)) GO TO 34
536     45 CONTINUE
540        GO TO 36
541    239 IF(NR .EQ.0) GO TO 36
544        DO 240 I = 1,NR
545    240 IF(REGNOS(I).EQ. QRN(INDEX)) GO TO 334
551        GO TO 36
552    334 IF(REGNWT(I) .GE. RNWT(INDEX)) GO TO 34
555        GO TO 36
556    147 IF(NGEN(QN).EQ.TERM(1)) GO TO 151
561        IF(NGEN(QN).EQ.TERM(2)) GO TO 152
564        IF(NGEN(QN).EQ.TERM(3)) GO TO 153
567        IF(NGEN(QN).EQ.TERM(4)) GO TO 154
572        WRITE(6,333)
573    333 FORMAT (1H0,20HILLEGAL GENERIC TERM)
574        GO TO 36
575    151 N = BARBIT(1)
576        DO 155 I=2,N
577        DO 155 J=1,NR
600    155 IF(REGNOS(J).EQ.BARBIT(I)) GO TO 34
605        GO TO 36
606    152 N = HYDANT(1)
607        DO 156 I=2,N
610        DO 156 J=1,NR
611    156 IF (REGNOS(J) .EQ. HYDANT(I)) GO TO 34
616        GO TO 36
617    153 N = OXAZOL(1)
620        DO 157 I=2,N
621        DO 157 J=1,NR
622    157 IF (REGNOS(J) .EQ. OXAZOL(I)) GO TO 34
627        GO TO 36
630    154 N = SUCCIN(1)
631        DO 158 I=2,N
632        DO 158 J=1,NR
633    158 IF (REGNOS(J) .EQ. SUCCIN(I)) GO TO 34
640        GO TO 36
     C ** END OF SEARCH ROUTINE **
641     53 CC = CC+1
642        IF(CC.EQ.73) GO TO 192
645        IF (LOGEXP(CC).EQ.BLANK) GO TO 53
650        IF (LOGEXP(CC).EQ.PERIOD.OR.LOGEXP(CC).EQ.RP) GO TO 55
```

```
ISN       SOURCE STATEMENT

653          IF (LOGEXP(CC).EQ.DNA) GO TO 54
656          IF (QTV*M) 49,99,56
657       54 IF (QTV*M) 57,99,49
660       55 IF (QTV*M) 57,99,56
      C SET EXPRESSION JUST CHECKED TO TRUE.
661       56 EXV = 1
662          GO TO 58
      C SET EXPRESSION JUST CHECKED TO FALSE.
663       57 EXV = 2
      C ****** END OF EVAL ROUTINE  ****
664       58 JMP = EXV+2*IND
665          GO TO (86,86,76,75,63,62), JMP
666       76 ICOUNT= NSAT(QN)+1
      C SAVE ARTICLE NUMBER FOR LATER PRINTING.
667          SATIS(QN,ICOUNT) = AN
670          NSAT(QN) = ICOUNT
671       75 CONTINUE
      C GO READ NEXT ARTICLE
673          GO TO 30
674       62 LOGEXP(CC2) = CHARST(26)
675          GO TO 64
676       63 LOGEXP(CC2) = CHARST(25)
677       64 CC = CC2-1
700          DO 65 J = CC3,CC
701       65 LOGEXP(J) = BLANK
703          IF (CC1.NE.CC3) GO TO 67
706          IND = 1
707      165 CC3 = CC3 -1
710          IF (CC3.EQ.0) GO TO 49
713          IF (LOGEXP(CC3).EQ.BLANK) GO TO 165
716          IF (LOGEXP(CC3).NE.MINUS) GO TO 49
721          IF(LOGEXP(CC2).NE.CHARST(25)) GO TO 166
724          LOGEXP (CC2) = CHARST(26)
725          GO TO 49
726      166 IF (LOGEXP(CC2).EQ.CHARST(26)) LOGEXP(CC2) = CHARST(25)
731          GO TO 49
      C RETURN FROM EVAL POINT 2.
      C SAVE LOCATION OF LEFTMOST PARENTHESIS.
732       66 CC1 = CC
      C SET INDICATOR SHOWING WITHIN PAREN.
733          IND = 2
734       67 CC = CC+1
735          IF (CC.GE.73) GO TO 96
740          IF (LOGEXP(CC).NE.RP) GO TO 67
      C SAVE LOCATION OF RIGHT PARENTHESIS.
743          CC2 = CC
744       68 CC = CC - 1
745          IF (LOGEXP(CC).NE.LP) GO TO 68
      C SAVE LOCATION OF LEFT PARENTHESIS.
750          CC3 = CC
751          GO TO 49
752       96 WRITE (6,97) (LOGX(QN,I),I=1,72)
757       97 FORMAT (23HOUNBALANCED PARENTHESIS/1H ,72A1)
760          GO TO 75
761       98 WRITE (6,198) LOGEXP(CC),CC,(LOGEXP(K),K=1,72)
```

```
ISN       SOURCE STATEMENT

766    198 FORMAT (10HOBAD CHAR  ,A1,5H LOC ,I3,2X,72A1)
767        GO TO 75
770     99 WRITE (6,199) CC,LOGEXP(CC)
771    199 FORMAT (10HOBAD TRV   ,I3,5X,A1)
772        CALL EXIT
773     81 WRITE (6,883)
774    883 FORMAT (1H1)
775        DO 82 I=1,NQ
776        K = NSAT(I)
777     82 WRITE (6,83) I,(SATIS(I,J),J=1,K)
1005    83 FORMAT(1HO,5HQUERY,I3,23H  SATISFIED BY ARTICLES,8I4,/(1H ,31X,8I4
          1))
1006       LST = 0
1007       DO 85 I = 1,NQ
1010       K = NSAT(I)
1011       IF (K.EQ.0) GO TO 88
1014       DO 84 J = 1,K
1015       LST = LST+1
1016    84 IS(LST) = SATIS(I,J)
1020    88 LST = LST+1
1021    85 IS(LST) = 0
1023       IS(LST+1) = -1
1024       CALL CHNXIT
1025   192 IERR=IERR+1
1026    92 IERR = IERR+1
1027       IDENT(NQ,LET,1) = 6
1030    89 IERR = IERR+1
1031    90 IERR = IERR+1
1032    91 IERR = IERR+1
.033    93 IERR = IERR+1
.034    94 WRITE (6,95) IERR,NQ, INPUT
.035    95 FORMAT (6HOERROR,I3,6H QUERY,I3,5X,12A6)
.036       SSW2=1
.037       WRITE(6,777)SSW1,SSW2,PACKNO
.040   777 FORMAT(1HO,3I5)
.041       GO TO 1
.042   100 WRITE (6,101) QN,TT,I
.043   101 FORMAT (9HOQUERY NO,I3,4H TT=,I4,3H I=,I3)
.044       WRITE(6,778)INPUT
.045   778 FORMAT(1HO,12A6)
.046       GO TO 75
.047    86 WRITE (6,87) IND,EXV,NQ,AN
.050    87 FORMAT (5HOIND=,I3,2X,4HEXV=,2X,3HQN=,I3,2X,3HAN=,I3)
.051       GO TO 75
.052       END
```

7. Print Program Listing

571 BAXEND FORTRAN SOURCE LIST

```
ISN       SOURCE STATEMENT

  0 $IBFTC PRINT
  1       INTEGER PR,QA,EOA,BUFFER(12),S(500),FILE(12,1150),EXCESS(20),
         1 DISK(50,3),TRACK
  2       DATA EOA/6H*END 0/
  3       COMMON S,LST
  4       COMMON /GC/ FILE,IFI,EXCESS,IEX,DISK,IDI,LOC(50,2),ILO,IS,EOA
    C AN I PREFIX ON A VARIABLE REFERS TO AN INDEX OF AN ARRAY
    C   IDENTIFIED BY THE STEM OF THAT VARIABLE.
  5       REWIND 4
  6       NQ = 1
    C SET PRINT OPTION OFF.  (1=OFF, 2=ON)
  7       PR = 1
    C INITIALIZE ALL INDICIES.
 10       IS = 0
 11       IDI = 0
 12       IFI = 0
 13       ILO = 0
 14       IEX = 0
 15       NA = 0
 16     1 IS = IS+1
 17       IF (S(IS)) 2,3,4
 20     2 REWIND 4
 21       CALL CHNXIT
 22     3 NQ = NQ+1
 23       GO TO 1
 24     4 ISP1 = IS + 1
 25       WRITE (6,5) NQ
 26     5 FORMAT (1H1,30X,12HQUERY NUMBER,I3)
 27       QA = S(IS)
 30    25 IF (QA-NA-1) 6,17,18
 31     6 DO 7 I = 1,ILO
 32       IF (QA-LOC(I,1)) 7,13,7
 33     7 CONTINUE
 35       DO 8 I = 1,IDI
 36       IF (QA-DISK(I,1)) 8,9,8
 37     8 CONTINUE
    C AT THIS POINT WE HAVE BEEN UNABLE TO LOCATE AN ARTICLE.  GO TO ERRORR.
 41       GO TO 900
 42     9 TRACK = DISK(I,2)
 43       NL = DISK(I,3)
 44       IF (IFI+NL.GT.1150) CALL SQUEEZ(NL)
 47       NWRD = NL*12
 50       M = IFI+1
    C UPDATE LOC.
 51       ILO = ILO+1
 52       LOC(ILO,1) = QA
 53       LOC(ILO,2) = M
 54       IFI = IFI+NL
    C SAVE STARTING LOCATION OF ARTICLE.
 55       N = M
 56    26 IF (NWRD.LE.456) GO TO 10
 61       CALL READR3(TRACK,456,FILE(1,M))
 62       TRACK = TRACK+1
 63       NWRD = NWRD-456
 64       M = M+38
```

571 BAXEND FORTRAN SOURCE LIST PRINT

```
ISN       SOURCE STATEMENT

 65          GO TO 26
 66       10 CALL READR3(TRACK,NWRD,FILE(1,M))
       C PRINT THE ARTICLE.
 67          DO 11 I = N,IFI
 70       11 WRITE (6,12) (FILE(J,I),J=1,12)
 76       12 FORMAT (1H ,12A6)
 77          GO TO 15
       C GET THE STARTING LOCATION AND PRINT THE ARTICLE.
100       13 M = LOC(I,2)-1
101       14 M = M+1
102          WRITE (6,12) (FILE(J,M),J=1,12)
107          IF (FILE(2,M).NE.EOA) GO TO 14
112       15 DO 16 I = ISP1,LST
113       16 IF (S(I).EQ.QA) GO TO 1
       C SINCE THERE ARE NO OTHER REFERENCES TO THIS ARTICE, PUT IT IN EXCESS.
117          IF (QA.EQ.LOC(ILO,1)) GO TO 27
122          IEX = IEX+1
123          EXCESS(IEX) = QA
124          GO TO 1
125       27 IFI = LOC(ILO,2)-1
126          ILO = ILO-1
127          GO TO 1
130       17 PR = 2
131       18 READ (4) NA
133          DO 19 I = ISP1,LST
134       19 IF (S(I).EQ.NA) GO TO 21
140       20 READ (4) BUFFER
142          IF (PR.EQ.2) WRITE (6,12) BUFFER
145          IF (BUFFER(2).NE.EOA) GO TO 20
150          GO TO 23
       C UPDATE LOC FOR NEW ARTICLE.
151       21 ILO = ILO+1
152          LOC(ILO,1) = NA
153          LOC(ILO,2) = IFI+1
154       22 IFI = IFI+1
155          IF (IFI.GT.1150) CALL SQUEEZ(50)
160          READ (4) (FILE(I,IFI),I=1,12)
165          IF (PR.EQ.2) WRITE (6,12) (FILE(I,IFI),I=1,12)
174          IF (FILE(2,IFI).NE.EOA) GO TO 22
       C CHECK PRINT OPTION.
177       23 GO TO (25,24), PR
200       24 PR = 1
201          GO TO 1
202      900 WRITE (6,901)
203      901 FORMAT (7HOERROR1)
204          REWIND 4
205          CALL CHNXIT
206          END
```

```
571                    BAXEND                    FORTRAN SOURCE LIST
      ISN       SOURCE STATEMENT

        0 $IBFTC SQEZ
        1       SUBROUTINE SQUEEZ(M)
        2       INTEGER FILE(12,1150),EXCESS(20),DISK(50,3),SUM,S(500),TEMP(50),
               1 EOA
        3       COMMON S,LST
        4       COMMON /GC/ FILE,IFI,EXCESS,IEX,DISK,IDI,LOC(50,2),ILO,IS,EOA
        5       SUM = 0
          C CHECK FOR NO EXCESS ARTICLES.
        6       IF (IEX.EQ.0) GO TO 15
          C COUNT THE NUMBER OF LINES IN EXCESS.
       11       DO 4 I = 1,IEX
       12       DO 1 J = 1,ILO
       13     1 IF (EXCESS(I).EQ.LOC( J,1)) GO TO 2
          C ERROR IF ARTICLE IN EXCESS IS NOT LOCATED IN -LOC-.
       17       WRITE (6,90)
       20       CALL EXIT
       21     2 IF (J.EQ.ILO) GO TO 3
       24       SUM = SUM+LOC(J+1,2)-LOC(J,2)
       25       GO TO 4
       26     3 SUM = SUM+IFI+1-LOC(J,2)
       27     4 CONTINUE
          C SEE HOW MUCH ROOM WOULD BE LEFT IN -FILE- IF EXCESS ARTICLES REMOVED.
       31     5 IF (IFI+M.GT.1150+SUM) GO TO 15
          C BEGIN OVERLAYING EXCESS ARTICLES.
          C LOCATE THE FIRST ARTICLE IN -FILE- IN EXCESS.
       34       DO 6 NLO = 1,ILO
       35       DO 6 J = 1,IEX
       36     6 IF (LOC(NLO,1).EQ.EXCESS(J)) GO TO 7
          C CHECK TO SEE IF FIRST ARTICLE IN EXCESS IS THE LAST ONE IN -FILE-.
       43     7 IF (NLO.EQ.ILO) GO TO 14
       46       NFI = LOC(NLO,2)
       47       NXT = NLO+1
       50       DO 11 I = NXT,ILO
          C IS THE NEXT ARTICLE IN EXCESS...
       51       DO 8 J = 1,IEX
       52     8 IF (LOC(I,1).EQ.EXCESS(J)) GO TO 11
       56       LOC(NLO,1) = LOC(I,1)
          C GET FIRST LINE OF ARTICLE.
       57       L = LOC(I,2)
       60       LOC(NLO,2) = NFI
          C GET LAST LINE OF ARTICLE.
       61       LL = LOC(I+1,2)-1
       62       IF (I.EQ.ILO) LL=IFI
          C MOVE ARTICLE UP IN -FILE-.
       65       DO 10 K = L,LL
       66       DO 9 J= 1,12
       67     9 FILE(J,NFI) = FILE(J,K)
       71    10 NFI = NFI+1
       73       NLO = NLO+1
       74    11 CONTINUE
          C READJUST PROGRAM PARAMETERS AND RETURN.
       76    13 IFI = NFI-1
       77       ILO = NLO-1
      100       IEX = 0
      101       RETURN
```

```
571                    BAXEND                    FORTRAN SOURCE LIST SQEZ
      ISN       SOURCE STATEMENT

          C DELETE ONLY THE LAST ARTICLE IN -FILE- AND RETURN.
      102    14 IFI = LOC(ILO,2)-1
      103       ILO = ILO-1
      104       IEX = 0
      105       RETURN
          C NOT ENOUGH ARTICLES IN EXCESS, SO PUT ONE ON THE DISK.
      106    15 ILOM1 = ILO-1
      107       DO 16 I = 1,ILOM1
      110    16 TEMP (I) = LOC(I,1)
      112       ICOUNT = ILOM1-IEX
      113       IF (IEX.EQ.0) GO TO 18
      116       DO 17 I = 1,IEX
      117       DO 17 J = 1,ILOM1
      120    17 IF (TEMP(J).EQ.EXCESS(I)) TEMP(J) = -2
      125    18 DO 19 I = IS,LST
      126       DO 19 J = 1,ILOM1
      127       IF (S(I).NE.TEMP(J)) GO TO 19
      132       IF (ICOUNT.LE.1) GO TO 20
      135       ICOUNT = ICOUNT-1
      136       TEMP(J) = -2
      137    19 CONTINUE
      142       WRITE (6,91)
      143       CALL EXIT
          C SEE IF THIS ARTICLE IS ALREADY STORED ON THE DISK.
      144    20 IF (IDI.EQ.0) GO TO 22
      147       DO 21 I = 1,IDI
      150    21 IF (TEMP(J).EQ.DISK(I,1)) GO TO 27
      154    22 NL = LOC(J+1,2)-LOC(J,2)
      155       SUM = SUM+NL
      156    24 IDI = IDI+1
          C FIND NEXT AVAILABLE DISK TRACK.
      157       NT = NT+1
      160       DISK(IDI,1) = TEMP(J)
      161       DISK(IDI,2) = NT
      162       DISK(IDI,3) = NL
      163       L = LOC(J,2)
          C SEE IF ARTICLE EXCEEDS 38 LINES.
      164    25 IF (NL.LE.38) GO TO 26
      167       CALL WRITR3(NT,456,FILE(1,L))
      170       NL = NL-38
      171       NT = NT+1
      172       L = L+38
      173       GO TO 25
      174    26 NWRD = NL*12
      175       CALL WRITR3(NT,NWRD,FILE(1,L))
      176       GO TO 28
      177    27 SUM = SUM+DISK(I,3)
      200    28 IEX = IEX+1
      201       EXCESS(IEX) = TEMP(J)
      202       GO TO 5
          C ERROR MESSAGES.
      203    90 FORMAT (44HOUNABLE TO LOCATE AN EXCESS ARTICLE IN -LOC-)
      204    91 FORMAT (42HOMORE ARTICLES IN -LOC- THAN REMAIN IN -S-)
      205       END
```